# GigaDevice Semiconductor Inc.

# GD32H73x_75x
# Arm® Cortex®-M7 32-bit MCU

For GD32H737xx, GD32H757xx, GD32H759xx, GD32H75Exx

# User Manual

Revision 1.8

(Mar. 2026)

# Table of Contents

**Table of Contents** ..... **2**

**List of Figures** ..... **40**

**List of Tables** ..... **54**

**1. System and memory architecture** ..... **61**

**1.1. Arm® Cortex®-M7 processor** ..... **61**

**1.2. System architecture** ..... **62**

1.2.1. Bus matrix Region 0 ..... 66

1.2.2. Bus matrix Region 1 ..... 66

1.2.3. Bus matrix Region 2 ..... 67

**1.3. Memory map** ..... **68**

1.3.1. On-chip SRAM memory ..... 83

1.3.2. On-chip flash memory overview ..... 86

**1.4. Boot configuration** ..... **86**

**1.5. System configuration controller (SYSCFG)** ..... **88**

**1.6. Timer break input lock** ..... **88**

**1.7. AXI interconnect matrix (AXIIM)** ..... **88**

1.7.1. Characteristics ..... 89

1.7.2. Function overview ..... 89

**1.8. System configuration registers** ..... **91**

1.8.1. Peripheral mode configuration register (SYSCFG_PMCFG) ..... 91

1.8.2. EXTI sources selection register 0 (SYSCFG_EXTISS0) ..... 93

1.8.3. EXTI sources selection register 1 (SYSCFG_EXTISS1) ..... 94

1.8.4. EXTI sources selection register 2 (SYSCFG_EXTISS2) ..... 95

1.8.5. EXTI sources selection register 3 (SYSCFG_EXTISS3) ..... 96

1.8.6. Lockup control register (SYSCFG_LKCTL) ..... 98

1.8.7. I/O compensation control register (SYSCFG_CPSCTL) ..... 99

1.8.8. I/O compensation cell code configuration register (SYSCFG_CPSCCCFG) ..... 100

1.8.9. Timer input selection register 0 (SYSCFG_TIMERCISEL0) ..... 101

1.8.10. Timer input selection register 1 (SYSCFG_TIMERCISEL1) ..... 102

1.8.11. Timer input selection register 2 (SYSCFG_TIMERCISEL2) ..... 103

1.8.12. Timer input selection register 3 (SYSCFG_TIMERCISEL3) ..... 104

1.8.13. Timer input selection register 4 (SYSCFG_TIMERCISEL4) ..... 105

1.8.14. Timer input selection register 5 (SYSCFG_TIMERCISEL5) ..... 107

1.8.15. Timer input selection register 6 (SYSCFG_TIMERCISEL6) ..... 109

1.8.16. CPU ICACHE error status register(SYSCFG_CPUICAC) ..... 110

1.8.17. CPU DCACHE error status register (SYSCFG_CPUDCAC) ..... 111

1.8.18. FPU interrupt enable register (SYSCFG_FPUINTEN) ..... 111

1.8.19. SRAM configuration register 0 (SYSCFG_SRAMCFG0) ..... 112
1.8.20. SRAM configuration register 1 (SYSCFG_SRAMCFG1) ..... 113
1.8.21. TIMERx configuration register 0 (SYSCFG_TIMERxCFG0, x=0, 7) ..... 113
1.8.22. TIMERx configuration register 1 (SYSCFG_TIMERxCFG1, x=0, 7) ..... 115
1.8.23. TIMERx configuration register 2 (SYSCFG_TIMERxCFG2, x=0, 7) ..... 117
1.8.24. TIMERx configuration register 0 (SYSCFG_TIMERxCFG0, x=1, 2, 3, 4, 22, 23, 30, 31) ... 118
1.8.25. TIMERx configuration register 1 (SYSCFG_TIMERxCFG1, x=1, 2, 3, 4, 22, 23, 30, 31) .. 121
1.8.26. TIMERx configuration register 2 (SYSCFG_TIMERxCFG2, x=1, 2, 3, 4, 22, 23, 30, 31) .. 123
1.8.27. TIMERx configuration register 0 (SYSCFG_TIMERxCFG0, x=14, 40, 41, 42, 43, 44) ..... 124
1.8.28. TIMERx configuration register 1 (SYSCFG_TIMERxCFG1, x=14, 40, 41, 42, 43, 44) ..... 126
1.8.29. TIMERx configuration register 2 (SYSCFG_TIMERxCFG2, x=14, 40, 41, 42, 43, 44) ..... 127
1.8.30. User configuration register (SYSCFG_USERCFG) ..... 128

**1.9. AXI interconnect registers ..... 129**
1.9.1. AXI peripheral ID4 register (AXI_PERIPH_ID4) ..... 129
1.9.2. AXI peripheral ID0 register (AXI_PERIPH_ID0) ..... 130
1.9.3. AXI peripheral ID1 register (AXI_PERIPH_ID1) ..... 130
1.9.4. AXI peripheral ID2 register (AXI_PERIPH_ID2) ..... 130
1.9.5. AXI peripheral ID3 register (AXI_PERIPH_ID3) ..... 131
1.9.6. AXI componet ID0 register (AXI_COMP_ID0) ..... 131
1.9.7. AXI componet ID1 register (AXI_COMP_ID1) ..... 132
1.9.8. AXI componet ID2 register (AXI_COMP_ID2) ..... 132
1.9.9. AXI componet ID3 register (AXI_COMP_ID3) ..... 133
1.9.10. AXI Master Port x bus matrix issuing functionality control register (AXI_MPxBM_ISS_CTL) 133
1.9.11. AXI Master Port x bus matrix functionality control register (AXI_MPxBM_CTL) ..... 134
1.9.12. AXI Master Port x long burst functionality control register (AXI_MPx_LB_CTL) ..... 134
1.9.13. AXI Master Port x issuing functionality control register (AXI_MPx_ISS_CTL) ..... 135
1.9.14. AXI Slave Port x functionality control register (AXI_SPx_CTL) ..... 135
1.9.15. AXI Slave Port x AHB issuing functionality control register (AXI_SPx_AHBISS_CTL) ..... 136
1.9.16. AXI Slave Port x read QOS control register (AXI_SPx_RDQOS_CTL) ..... 136
1.9.17. AXI Slave Port x write QOS control register (AXI_SPx_WRQOS_CTL) ..... 137
1.9.18. AXI Slave Port x issuing functionality control register (AXI_SPx_ISS_CTL) ..... 137

**1.10. Device electronic signature ..... 138**
1.10.1. Memory density information ..... 138
1.10.2. Unique device ID (96 bits) ..... 138

**2. RAM ECC monitor unit (RAMECCMU) ..... 140**

**2.1. Characteristics ..... 140**

**2.2. Function overview ..... 140**

**2.3. Register definition ..... 142**
2.3.1. RAMECCMU global interruput register (RAMECCMU_INT) ..... 142
2.3.2. RAMECCMU monitor x control register (RAMECCMU_MxCTL) ..... 142
2.3.3. RAMECCMU monitor x status register (RAMECCMU_MxSTAT) ..... 143

2.3.4. RAMECCMU monitor x failing address register (RAMECCMU_MxFADDR) ..... 144
2.3.5. RAMECCMU monitor x failing data low register (RAMECCMU_MxFDL) ..... 144
2.3.6. RAMECCMU monitor x failing data high register (RAMECCMU_MxFDH) ..... 145
2.3.7. RAMECCMU monitor x failing ECC error code register (RAMECCMU_MxFECODE) ..... 145

**3. Flash memory controller (FMC) ..... 147**

**3.1. Overview ..... 147**

**3.2. Characteristics ..... 147**

**3.3. Function overview ..... 147**

3.3.1. Flash memory architecture ..... 147
3.3.2. Read operations ..... 148
3.3.3. Unlock the FMC_CTL and FMC_OBCTL register ..... 149
3.3.4. Sector erase ..... 150
3.3.5. Mass erase ..... 151
3.3.6. Main flash Programming ..... 156
3.3.7. Option bytes ..... 157
3.3.8. Sector erase/program protection ..... 161
3.3.9. Security protection ..... 162
3.3.10. Dedicated code read protection area ..... 163
3.3.11. Secure user area ..... 165
3.3.12. Secure mode ..... 167
3.3.13. Basic security service ..... 168
3.3.14. Error description ..... 169
3.3.15. FMC interrupts ..... 171

**3.4. Register definition ..... 172**

3.4.1. Unlock key register (FMC_KEY) ..... 172
3.4.2. Option byte unlock key register (FMC_OBKEY) ..... 172
3.4.3. Control register (FMC_CTL) ..... 172
3.4.4. Status register (FMC_STAT) ..... 174
3.4.5. Address register (FMC_ADDR) ..... 175
3.4.6. Option byte control register (FMC_OBCTL) ..... 176
3.4.7. Option byte status register 0 (FMC_OBSTAT0_EFT) ..... 176
3.4.8. Option byte status register 0 (FMC_OBSTAT0_MDF) ..... 178
3.4.9. DCRP address register (FMC_DCRPADDR_EFT) ..... 180
3.4.10. DCRP address register (FMC_DCRPADDR_MDF) ..... 181
3.4.11. Secure address register (FMC_SCRADDR_EFT) ..... 181
3.4.12. Secure address register (FMC_SCRADDR_MDF) ..... 182
3.4.13. Erase/program protection register (FMC_WP_EFT) ..... 183
3.4.14. Erase/program protection register (FMC_WP_MDF) ..... 184
3.4.15. Boot address register (FMC_BTADDR_EFT) ..... 185
3.4.16. Boot address register (FMC_BTADDR_MDF) ..... 185
3.4.17. Option byte status register 1 (FMC_OBSTAT1_EFT) ..... 186
3.4.18. Option byte status register 1 (FMC_OBSTAT1_MDF) ..... 186

3.4.19. NO-RTDEC area register (FMC_NODEC) ..... 187
3.4.20. AES IV register (FMC_AESIVx_EFT) (x = 0…2) ..... 188
3.4.21. AES IV register (FMC_AESIVx_MDF) (x = 0…2) ..... 189
3.4.22. Product ID register x (FMC_PIDx) (x = 0,1) ..... 189

**4. Electronic fuse (EFUSE) ..... 191**

**4.1. Overview ..... 191**

**4.2. Characteristics ..... 191**

**4.3. Function overview ..... 191**

4.3.1. Block diagram ..... 191
4.3.2. Efuse macro description ..... 192
4.3.3. Read operation ..... 195
4.3.4. Program operation ..... 196
4.3.5. AES key CRC function ..... 196
4.3.6. EFUSE interrupts ..... 197

**4.4. Register definition ..... 199**

4.4.1. Control register (EFUSE_CTL) ..... 199
4.4.2. Address register (EFUSE_ADDR) ..... 200
4.4.3. Status register (EFUSE_STAT) ..... 201
4.4.4. Status flag clear register (EFUSE_STATC) ..... 201
4.4.5. User control register (EFUSE_USER_CTL) ..... 202
4.4.6. MCU reserved register (EFUSE_MCU_RSV) ..... 205
4.4.7. Debug password register x (EFUSE_DPx) (x = 0,1) ..... 206
4.4.8. Firmware AES key register x (EFUSE_AES_KEYx) (x = 0…3) ..... 207
4.4.9. User data register x (EFUSE_USER_DATAx) (x = 0…3) ..... 208

**5. Power management unit (PMU) ..... 209**

**5.1. Overview ..... 209**

**5.2. Characteristics ..... 209**

**5.3. Function overview ..... 210**

5.3.1. Backup domain ..... 211
5.3.2. $V_{DD}$ / $V_{DDA}$ power domain ..... 212
5.3.3. $V_{CORE}$ power domain ..... 216
5.3.4. USB Power supply ..... 219
5.3.5. Power saving modes ..... 219

**5.4. Register definition ..... 222**

5.4.1. Control register 0 (PMU_CTL0) ..... 222
5.4.2. Control and status register (PMU_CS) ..... 223
5.4.3. Control register 1 (PMU_CTL1) ..... 225
5.4.4. Control register 2 (PMU_CTL2) ..... 227
5.4.5. Control register 3 (PMU_CTL3) ..... 228
5.4.6. Parameter register (PMU_PAR) ..... 229

**6. Reset and clock unit (RCU) ..... 231**

**6.1. Reset control unit (RCTL) ..... 231**

6.1.1. Overview ..... 231

6.1.2. Function overview ..... 231

**6.2. Clock control unit (CCTL) ..... 232**

6.2.1. Overview(For GD32H737_757_759) ..... 232

6.2.2. Overview(For GD32H75E) ..... 236

6.2.3. Characteristics ..... 239

6.2.4. Function overview ..... 240

**6.3. Register definition(For GD32H737_757_759) ..... 246**

6.3.1. Control register (RCU_CTL) ..... 246

6.3.2. PLL0 register (RCU_PLL0) ..... 248

6.3.3. Clock configuration register 0 (RCU_CFG0) ..... 250

6.3.4. Clock interrupt register (RCU_INT) ..... 252

6.3.5. AHB1 reset register (RCU_AHB1RST) ..... 255

6.3.6. AHB2 reset register (RCU_AHB2RST) ..... 257

6.3.7. AHB3 reset register (RCU_AHB3RST) ..... 258

6.3.8. AHB4 reset register (RCU_AHB4RST) ..... 259

6.3.9. APB1 reset register (RCU_APB1RST) ..... 261

6.3.10. APB2 reset register (RCU_APB2RST) ..... 264

6.3.11. APB3 reset register (RCU_APB3RST) ..... 267

6.3.12. APB4 reset register (RCU_APB4RST) ..... 268

6.3.13. AHB1 enable register (RCU_AHB1EN) ..... 268

6.3.14. AHB2 enable register (RCU_AHB2EN) ..... 270

6.3.15. AHB3 enable register (RCU_AHB3EN) ..... 272

6.3.16. AHB4 enable register (RCU_AHB4EN) ..... 273

6.3.17. APB1 enable register (RCU_APB1EN) ..... 275

6.3.18. APB2 enable register (RCU_APB2EN) ..... 278

6.3.19. APB3 enable register (RCU_APB3EN) ..... 281

6.3.20. APB4 enable register (RCU_APB4EN) ..... 282

6.3.21. AHB1 sleep mode enable register (RCU_AHB1SPEN) ..... 283

6.3.22. AHB2 sleep mode enable register (RCU_AHB2SPEN) ..... 285

6.3.23. AHB3 sleep mode enable register (RCU_AHB3SPEN) ..... 286

6.3.24. AHB4 sleep mode enable register (RCU_AHB4SPEN) ..... 288

6.3.25. APB1 sleep mode enable register (RCU_APB1SPEN) ..... 290

6.3.26. APB2 sleep mode enable register (RCU_APB2SPEN) ..... 293

6.3.27. APB3 sleep mode enable register (RCU_APB3SPEN) ..... 296

6.3.28. APB4 sleep mode enable register (RCU_APB4SPEN) ..... 297

6.3.29. Backup domain control register (RCU_BDCTL) ..... 298

6.3.30. Reset source/clock register (RCU_RSTSCK) ..... 299

6.3.31. PLL clock additional control register (RCU_PLLADDCTL) ..... 301

6.3.32. PLL1 register (RCU_PLL1) ..... 303

6.3.33. PLL2 register (RCU_PLL2) ..... 305

6.3.34. Clock configuration register 1 (RCU_CFG1) ..... 307
6.3.35. Clock configuration register 2 (RCU_CFG2) ..... 309
6.3.36. Clock configuration register 3 (RCU_CFG3) ..... 312
6.3.37. PLL all configuration register (RCU_PLLALL) ..... 314
6.3.38. PLL0 fraction configuration register (RCU_PLL0FRA) ..... 315
6.3.39. PLL1 fraction configuration register (RCU_PLL1FRA) ..... 316
6.3.40. PLL2 fraction configuration register (RCU_PLL2FRA) ..... 316
6.3.41. Additional clock control register 0 (RCU_ADDCTL0) ..... 317
6.3.42. Additional clock control register 1(RCU_ADDCTL1) ..... 318
6.3.43. Additional clock interrupt register (RCU_ADDINT) ..... 320
6.3.44. Clock configuration register 4 (RCU_CFG4) ..... 321
6.3.45. USB clock control register (RCU_USBCLKCTL) ..... 322
6.3.46. PLLUSB configuration register (RCU_PLLUSBCFG) ..... 324
6.3.47. APB2 additional reset register (RCU_ADDAPB2RST) ..... 326
6.3.48. APB2 additional enable register (RCU_ADDAPB2EN) ..... 326
6.3.49. APB2 additional sleep enable register (RCU_ADDAPB2SPEN) ..... 327
6.3.50. Clock configuration register 5 (RCU_CFG5) ..... 328

**6.4. Register definition(For GD32H75E) ..... 330**
**6.4.1. Control register (RCU_CTL) ..... 330**
**6.4.2. PLL0 register (RCU_PLL0) ..... 332**
**6.4.3. Clock configuration register 0 (RCU_CFG0) ..... 334**
**6.4.4. Clock interrupt register (RCU_INT) ..... 336**
**6.4.5. AHB1 reset register (RCU_AHB1RST) ..... 339**
**6.4.6. AHB2 reset register (RCU_AHB2RST) ..... 340**
**6.4.7. AHB3 reset register (RCU_AHB3RST) ..... 341**
**6.4.8. AHB4 reset register (RCU_AHB4RST) ..... 342**
**6.4.9. APB1 reset register (RCU_APB1RST) ..... 343**
**6.4.10. APB2 reset register (RCU_APB2RST) ..... 346**
**6.4.11. APB3 reset register (RCU_APB3RST) ..... 349**
**6.4.12. APB4 reset register (RCU_APB4RST) ..... 350**
**6.4.13. AHB1 enable register (RCU_AHB1EN) ..... 351**
**6.4.14. AHB2 enable register (RCU_AHB2EN) ..... 352**
**6.4.15. AHB3 enable register (RCU_AHB3EN) ..... 353**
**6.4.16. AHB4 enable register (RCU_AHB4EN) ..... 354**
**6.4.17. APB1 enable register (RCU_APB1EN) ..... 355**
**6.4.18. APB2 enable register (RCU_APB2EN) ..... 358**
**6.4.19. APB3 enable register (RCU_APB3EN) ..... 361**
**6.4.20. APB4 enable register (RCU_APB4EN) ..... 362**
**6.4.21. AHB1 sleep mode enable register (RCU_AHB1SPEN) ..... 362**
**6.4.22. AHB2 sleep mode enable register (RCU_AHB2SPEN) ..... 364**
**6.4.23. AHB3 sleep mode enable register (RCU_AHB3SPEN) ..... 365**
**6.4.24. AHB4 sleep mode enable register (RCU_AHB4SPEN) ..... 366**
**6.4.25. APB1 sleep mode enable register (RCU_APB1SPEN) ..... 368**
**6.4.26. APB2 sleep mode enable register (RCU_APB2SPEN) ..... 371**

6.4.27. APB3 sleep mode enable register (RCU_APB3SPEN) ..... 373
6.4.28. APB4 sleep mode enable register (RCU_APB4SPEN) ..... 374
6.4.29. Backup domain control register (RCU_BDCTL) ..... 375
6.4.30. Reset source/clock register (RCU_RSTSCK) ..... 377
6.4.31. PLL clock additional control register (RCU_PLLADDCTL) ..... 378
6.4.32. PLL1 register (RCU_PLL1) ..... 381
6.4.33. PLL2 register (RCU_PLL2) ..... 382
6.4.34. Clock configuration register 1 (RCU_CFG1) ..... 384
6.4.35. Clock configuration register 2 (RCU_CFG2) ..... 386
6.4.36. Clock configuration register 3 (RCU_CFG3) ..... 388
6.4.37. PLL all configuration register (RCU_PLLALL) ..... 389
6.4.38. PLL0 fraction configuration register (RCU_PLL0FRA) ..... 391
6.4.39. PLL1 fraction configuration register (RCU_PLL1FRA) ..... 392
6.4.40. PLL2 fraction configuration register (RCU_PLL2FRA) ..... 392
6.4.41. Additional clock control register 0 (RCU_ADDCTL0) ..... 393
6.4.42. Additional clock control register 1(RCU_ADDCTL1) ..... 394
6.4.43. Additional clock interrupt register (RCU_ADDINT) ..... 396
6.4.44. Clock configuration register 4 (RCU_CFG4) ..... 397
6.4.45. USB clock control register (RCU_USBCLKCTL) ..... 398
6.4.46. PLLUSB configuration register (RCU_PLLUSBCFG) ..... 400
6.4.47. APB2 additional reset register (RCU_ADDAPB2RST) ..... 402
6.4.48. APB2 additional enable register (RCU_ADDAPB2EN) ..... 402
6.4.49. APB2 additional sleep enable register (RCU_ADDAPB2SPEN) ..... 403
6.4.50. Clock configuration register 5 (RCU_CFG5) ..... 404

**7. Clock trim controller (CTC) ..... 406**

**7.1. Overview ..... 406**

**7.2. Characteristics ..... 406**

**7.3. Function overview ..... 406**

7.3.1. REF sync pulse generator ..... 407
7.3.2. CTC trim counter ..... 407
7.3.3. Frequency evaluation and automatically trim process ..... 408
7.3.4. Software program guide ..... 409

**7.4. Register definition ..... 411**

7.4.1. Control register 0 (CTC_CTL0) ..... 411
7.4.2. Control register 1 (CTC_CTL1) ..... 412
7.4.3. Status register (CTC_STAT) ..... 413
7.4.4. Interrupt clear register (CTC_INTC) ..... 415

**8. Interrupt / event controller (EXTI) ..... 417**

**8.1. Overview ..... 417**

**8.2. Characteristics ..... 417**

**8.3. Interrupts function overview ..... 417**

8.4. External interrupt and event (EXTI) block diagram ..... 429
8.5. External interrupt and event function overview ..... 429
8.6. Register definition ..... 433
8.6.1. Interrupt enable register 0 (EXTI_INTEN0) ..... 433
8.6.2. Event enable register 0 (EXTI_EVEN0) ..... 433
8.6.3. Rising edge trigger enable register 0 (EXTI_RTEN0) ..... 433
8.6.4. Falling edge trigger enable register 0 (EXTI_FTEN0) ..... 434
8.6.5. Software interrupt event register 0 (EXTI_SWIEV0) ..... 434
8.6.6. Pending register 0 (EXTI_PD0) ..... 435
8.6.7. Interrupt enable register 1 (EXTI_INTEN1) ..... 435
8.6.8. Event enable register 1 (EXTI_EVEN1) ..... 436
8.6.9. Rising edge trigger enable register 1 (EXTI_RTEN1) ..... 436
8.6.10. Falling edge trigger enable register 1 (EXTI_FTEN1) ..... 436
8.6.11. Software interrupt event register 1 (EXTI_SWIEV1) ..... 437
8.6.12. Pending register 1 (EXTI_PD1) ..... 437
9. Trigger selection controller (TRIGSEL) ..... 439
9.1. Overview ..... 439
9.2. Characteristics ..... 439
9.3. Function overview ..... 439
9.4. Internal connect ..... 440
9.5. Register definition ..... 450
9.5.1. Trigger selection for EXTOUT register 0 (TRIGSEL_EXTOUT_0) ..... 450
9.5.2. Trigger selection for EXTOUT register 1 (TRIGSEL_EXTOUT_1) ..... 450
9.5.3. Trigger selection for EXTOUT register 2 (TRIGSEL_EXTOUT_2) ..... 451
9.5.4. Trigger selection for EXTOUT register 3 (TRIGSEL_EXTOUT_3) ..... 452
9.5.5. Trigger selection for ADC0 register (TRIGSEL_ADC0) ..... 452
9.5.6. Trigger selection for ADC1 register (TRIGSEL_ADC1) ..... 453
9.5.7. Trigger selection for ADC2 register (TRIGSEL_ADC2) ..... 454
9.5.8. Trigger selection for DAC0_OUT0 register (TRIGSEL_DAC0OUT0) ..... 454
9.5.9. Trigger selection for DAC0_OUT1 register (TRIGSEL_DAC0OUT1) ..... 455
9.5.10. Trigger selection for TIMER0_BRKIN register (TRIGSEL_TIMER0BRKIN) ..... 456
9.5.11. Trigger selection for TIMER7_BRKIN register (TRIGSEL_TIMER7BRKIN) ..... 456
9.5.12. Trigger selection for TIMER14_BRKIN register (TRIGSEL_TIMER14BRKIN) ..... 457
9.5.13. Trigger selection for TIMER15_BRKIN register (TRIGSEL_TIMER15BRKIN) ..... 458
9.5.14. Trigger selection for TIMER16_BRKIN register (TRIGSEL_TIMER16BRKIN) ..... 459
9.5.15. Trigger selection for TIMER40_BRKIN register (TRIGSEL_TIMER40BRKIN) ..... 459
9.5.16. Trigger selection for TIMER41_BRKIN register (TRIGSEL_TIMER41BRKIN) ..... 460
9.5.17. Trigger selection for TIMER42_BRKIN register (TRIGSEL_TIMER42BRKIN) ..... 460
9.5.18. Trigger selection for TIMER43_BRKIN register (TRIGSEL_TIMER43BRKIN) ..... 461
9.5.19. Trigger selection for TIMER44_BRKIN register (TRIGSEL_TIMER44BRKIN) ..... 462
9.5.20. Trigger selection for CAN0 register (TRIGSEL_CAN0) ..... 462

9.5.21. Trigger selection for CAN1 register (TRIGSEL_CAN1) ..... 463
9.5.22. Trigger selection for CAN2 register (TRIGSEL_CAN2) ..... 464
9.5.23. Trigger selection for LPDTS register (TRIGSEL_LPDTS) ..... 464
9.5.24. Trigger selection for TIMER0_ETI register (TRIGSEL_TIMER0ETI) ..... 465
9.5.25. Trigger selection for TIMER1_ETI register (TRIGSEL_TIMER1ETI) ..... 465
9.5.26. Trigger selection for TIMER2_ETI register (TRIGSEL_TIMER2ETI) ..... 466
9.5.27. Trigger selection for TIMER3_ETI register (TRIGSEL_TIMER3ETI) ..... 467
9.5.28. Trigger selection for TIMER4_ETI register (TRIGSEL_TIMER4ETI) ..... 467
9.5.29. Trigger selection for TIMER7_ETI register (TRIGSEL_TIMER7ETI) ..... 468
9.5.30. Trigger selection for TIMER22_ETI register (TRIGSEL_TIMER22ETI) ..... 468
9.5.31. Trigger selection for TIMER23_ETI register (TRIGSEL_TIMER23ETI) ..... 469
9.5.32. Trigger selection for TIMER30_ETI register (TRIGSEL_TIMER30ETI) ..... 470
9.5.33. Trigger selection for TIMER31_ETI register (TRIGSEL_TIMER31ETI) ..... 470
9.5.34. Trigger selection for EDOUT register (TRIGSEL_EDOUT) ..... 471
9.5.35. Trigger selection for HPDF register (TRIGSEL_HPDF) ..... 471
9.5.36. Trigger selection for TIMER0_ITI14 register (TRIGSEL_TIMER0ITI14) ..... 472
9.5.37. Trigger selection for TIMER1_ITI14 register (TRIGSEL_TIMER1ITI14) ..... 473
9.5.38. Trigger selection for TIMER2_ITI14 register (TRIGSEL_TIMER2ITI14) ..... 473
9.5.39. Trigger selection for TIMER3_ITI14 register (TRIGSEL_TIMER3ITI14) ..... 474
9.5.40. Trigger selection for TIMER4_ITI14 register (TRIGSEL_TIMER4ITI14) ..... 474
9.5.41. Trigger selection for TIMER7_ITI14 register (TRIGSEL_TIMER7ITI14) ..... 475
9.5.42. Trigger selection for TIMER14_ITI14 register (TRIGSEL_TIMER14ITI14) ..... 476
9.5.43. Trigger selection for TIMER22_ITI14 register (TRIGSEL_TIMER22ITI14) ..... 476
9.5.44. Trigger selection for TIMER23_ITI14 register (TRIGSEL_TIMER23ITI14) ..... 477
9.5.45. Trigger selection for TIMER30_ITI14 register (TRIGSEL_TIMER30ITI14) ..... 477
9.5.46. Trigger selection for TIMER31_ITI14 register (TRIGSEL_TIMER31ITI14) ..... 478
9.5.47. Trigger selection for TIMER40_ITI14 register (TRIGSEL_TIMER40ITI14) ..... 479
9.5.48. Trigger selection for TIMER41_ITI14 register (TRIGSEL_TIMER41ITI14) ..... 479
9.5.49. Trigger selection for TIMER42_ITI14 register (TRIGSEL_TIMER42ITI14) ..... 480
9.5.50. Trigger selection for TIMER43_ITI14 register (TRIGSEL_TIMER43ITI14) ..... 480
9.5.51. Trigger selection for TIMER44_ITI14 register (TRIGSEL_TIMER44ITI14) ..... 481

**10. General-purpose and alternate-function I/Os (GPIO and AFIO) ..... 482**

**10.1. Overview ..... 482**

**10.2. Characteristics ..... 482**

**10.3. Function overview ..... 482**

10.3.1. GPIO pin configuration ..... 484
10.3.2. External interrupt/event lines ..... 484
10.3.3. Alternate functions (AF) ..... 484
10.3.4. Additional functions ..... 484
10.3.5. Input configuration ..... 485
10.3.6. Output configuration ..... 485
10.3.7. Analog configuration ..... 486
10.3.8. Alternate function (AF) configuration ..... 486

10.3.9. GPIO locking function ..... 487
10.3.10. GPIO single cycle toggle function ..... 487
10.3.11. I/O compensation unit ..... 487
10.3.12. Analog configuration for ADC ..... 488
10.3.13. Input filtering ..... 488
**10.4. Register definition ..... 492**
10.4.1. Port control register (GPIOx_CTL, x=A...H, J, K) ..... 492
10.4.2. Port output mode register (GPIOx_OMODE, x=A…H, J, K) ..... 494
10.4.3. Port output speed register (GPIOx_OSPD, x=A...H, J, K) ..... 495
10.4.4. Port pull-up/down register (GPIOx_PUD, x=A…H, J, K) ..... 497
10.4.5. Port input status register (GPIOx_ISTAT, x=A…H, J, K) ..... 499
10.4.6. Port output control register (GPIOx_OCTL, x=A…H, J, K) ..... 499
10.4.7. Port bit operate register (GPIOx_BOP, x=A...H, J, K) ..... 500
10.4.8. Port configuration lock register (GPIOx_LOCK, x=A...H, J, K) ..... 500
10.4.9. Alternate function selected register 0 (GPIOx_AFSEL0, x=A...H, J, K) ..... 501
10.4.10. Alternate function selected register 1 (GPIOx_AFSEL1, x=A…H, J, K) ..... 502
10.4.11. Bit clear register (GPIOx_BC, x=A…H, J, K) ..... 503
10.4.12. Port bit toggle register (GPIOx_TG, x=A…H, J, K) ..... 504
10.4.13. Input filtering register (GPIOx_IFL, x=A...H, J, K) ..... 504
10.4.14. Input filtering type register (GPIOx_IFTP, x=A...H, J, K) ..... 505
**11. Cyclic redundancy checks management unit (CRC) ..... 508**
**11.1. Overview ..... 508**
**11.2. Characteristics ..... 508**
**11.3. Function overview ..... 509**
**11.4. Register definition ..... 511**
11.4.1. Data register (CRC_DATA) ..... 511
11.4.2. Free data register (CRC_FDATA) ..... 511
11.4.3. Control register (CRC_CTL) ..... 512
11.4.4. Initialization data register (CRC_IDATA) ..... 512
11.4.5. Polynomial register (CRC_POLY) ..... 513
**12. True random number generator (TRNG) ..... 514**
**12.1. Overview ..... 514**
**12.2. Characteristics ..... 514**
**12.3. Function overview ..... 515**
12.3.1. LFSR ..... 515
12.3.2. Post processing ..... 516
12.3.3. Conditioning ..... 516
12.3.4. Output FIFO ..... 516
12.3.5. Health tests ..... 516
12.3.6. NIST mode state ..... 517
12.3.7. Operation flow ..... 518

12.3.8. Error flags ..... 519
12.3.9. Low power usage ..... 519
**12.4. Register definition ..... 520**
12.4.1. Control register (TRNG_CTL) ..... 520
12.4.2. Status register (TRNG_STAT) ..... 522
12.4.3. Data register (TRNG_DATA) ..... 523
12.4.4. Health tests configure register (TRNG_HTCFG) ..... 523
**13. Cryptographic Acceleration Unit (CAU) (For GD32H737_757_759) ..... 525**
**13.1. Overview ..... 525**
**13.2. Characteristics ..... 525**
**13.3. CAU data type and initialization vectors ..... 526**
13.3.1. Data type ..... 526
13.3.2. Initialization vectors ..... 527
**13.4. Cryptographic acceleration processor ..... 527**
13.4.1. DES / TDES cryptographic acceleration processor ..... 528
13.4.2. AES cryptographic acceleration processor ..... 532
**13.5. Operating modes ..... 540**
**13.6. CAU DMA interface ..... 542**
**13.7. CAU interrupts ..... 542**
**13.8. CAU suspended mode ..... 543**
**13.9. Register definition ..... 544**
13.9.1. Control register (CAU_CTL) ..... 544
13.9.2. Status register 0 (CAU_STAT0) ..... 546
13.9.3. Data input register (CAU_DI) ..... 546
13.9.4. Data output register (CAU_DO) ..... 547
13.9.5. DMA enable register (CAU_DMAEN) ..... 547
13.9.6. Interrupt enable register (CAU_INTEN) ..... 548
13.9.7. Status register 1 (CAU_STAT1) ..... 548
13.9.8. Interrupt flag register (CAU_INTF) ..... 549
13.9.9. Key registers (CAU_KEY0..3(H / L)) ..... 550
13.9.10. Initial vector registers (CAU_IV0..1(H / L)) ..... 552
13.9.11. GCM or CCM mode context switch register x (CAU_GCMCCMCTXSx) (x=0..7) ..... 554
13.9.12. GCM mode context switch register x (CAU_GCMCTXSx) (x=0..7) ..... 554
**14. Hash Acceleration Unit (HAU) (For GD32H737_757_759) ..... 556**
**14.1. Overview ..... 556**
**14.2. Characteristics ..... 556**
**14.3. HAU data type ..... 556**
**14.4. HAU core ..... 558**

14.4.1. Automatic data padding ..... 558
14.4.2. Digest computing ..... 559
14.4.3. Hash mode ..... 560
14.4.4. HMAC mode ..... 560
**14.5. HAU suspended mode ..... 560**
14.5.1. Transfer data by CPU ..... 561
14.5.2. Transfer data by DMA ..... 561
**14.6. HAU interrupt ..... 562**
14.6.1. Input FIFO interrupt ..... 562
14.6.2. Calculation completion interrupt ..... 562
**14.7. Register definition ..... 563**
14.7.1. Control register (HAU_CTL) ..... 563
14.7.2. Data input register (HAU_DI) ..... 564
14.7.3. Configuration register (HAU_CFG) ..... 565
14.7.4. Data output register (HAU_DO0..7) ..... 566
14.7.5. Interrupt enable register (HAU_INTEN) ..... 568
14.7.6. Status and flag register (HAU_STAT) ..... 569
14.7.7. Context switch register x (HAU_CTXSx) (x = 0...53) ..... 569
**15. Trigonometric Math Unit (TMU) ..... 571**
**15.1. Overview ..... 571**
**15.2. Characteristics ..... 571**
**15.3. Block diagram ..... 571**
**15.4. Function overview ..... 572**
15.4.1. Data format and configuration ..... 572
15.4.2. Mode configuration ..... 573
15.4.3. TMU precision ..... 582
15.4.4. TMU operation pending ..... 583
15.4.5. Zero-overhead mode ..... 583
15.4.6. Interrupt and DMA requests ..... 584
**15.5. Registers definition ..... 585**
15.5.1. Control and status register (TMU_CS) ..... 585
15.5.2. Input data register (TMU_IDATA) ..... 587
15.5.3. Output data register (TMU_ODATA) ..... 588
**16. Direct memory access controller (DMA) ..... 589**
**16.1. Overview ..... 589**
**16.2. Characteristics ..... 589**
**16.3. Function overview ..... 590**
16.3.1. Block diagram ..... 590
16.3.2. Peripheral handshake ..... 592

16.3.3. Data process ..... 592
16.3.4. Address generation ..... 597
16.3.5. Circular mode ..... 598
16.3.6. Switch-buffer mode ..... 598
16.3.7. Transfer operation ..... 599
16.3.8. Transfer finish ..... 600
16.3.9. Channel configuration ..... 601
**16.4. Interrupts ..... 602**
16.4.1. Flag ..... 603
16.4.2. Exception ..... 603
16.4.3. Error ..... 604
16.4.4. DMA request mapping ..... 606
**16.5. Register definition ..... 607**
16.5.1. Interrupt flag register 0 (DMA_INTF0) ..... 607
16.5.2. Interrupt flag register 1 (DMA_INTF1) ..... 608
16.5.3. Interrupt flag clear register 0 (DMA_INTC0) ..... 609
16.5.4. Interrupt flag clear register 1 (DMA_INTC1) ..... 610
16.5.5. Channel x control register (DMA_CHxCTL) ..... 610
16.5.6. Channel x counter register (DMA_CHxCNT) ..... 614
16.5.7. Channel x peripheral base address register (DMA_CHxPADDR) ..... 614
16.5.8. Channel x memory 0 base address register (DMA_CHxM0ADDR) ..... 615
16.5.9. Channel x memory 1 base address register (DMA_CHxM1ADDR) ..... 615
16.5.10. Channel x FIFO control register (DMA_CHxFCTL) ..... 616
**17. Master direct memory access controller (MDMA) ..... 618**
**17.1. Overview ..... 618**
**17.2. Characteristics ..... 618**
**17.3. Function overview ..... 619**
17.3.1. Data process ..... 621
17.3.2. Address generation ..... 624
17.3.3. Transfer modes ..... 625
17.3.4. Transfer status ..... 627
17.3.5. MDMA interrupts and errors ..... 628
**17.4. Register definition ..... 630**
17.4.1. Global interrupt flag register (MDMA_GINTF) ..... 630
17.4.2. Channel x status register 0 (MDMA_CHxSTAT0) ..... 630
17.4.3. Channel x status clear register (MDMA_CHxSTATC) ..... 631
17.4.4. Channel x status register 1 (MDMA_CHxSTAT1) ..... 632
17.4.5. Channel x control register 0 (MDMA_CHxCTL0) ..... 633
17.4.6. Channel x configure register (MDMA_CHxCFG) ..... 635
17.4.7. Channel x block transfer configure register (MDMA_CHxBTCFG) ..... 638
17.4.8. Channel x source address register (MDMA_CHxSADDR) ..... 639
17.4.9. Channel x destination address register (MDMA_CHxDADDR) ..... 640

17.4.10. Channel x multi-block address update register (MDMA_CHxMBADDRU) ..... 640
17.4.11. Channel x link address register (MDMA_CHxLADDR) ..... 641
17.4.12. Channel x control register 1 (MDMA_CHxCTL1) ..... 641
17.4.13. Channel x mask address register (MDMA_CHxMADDR) ..... 642
17.4.14. Channel x mask data register (MDMA_CHxMDATA) ..... 642
**18. DMA request multiplexer (DMAMUX) ..... 644**
**18.1. Overview ..... 644**
**18.2. Characteristics ..... 644**
**18.3. Block diagram ..... 645**
**18.4. Signal description ..... 645**
**18.5. Function overview ..... 646**
18.5.1. DMAMUX request multiplexer ..... 646
18.5.2. DMAMUX request generator ..... 649
18.5.3. Channel configurations ..... 649
18.5.4. Interrupt ..... 650
18.5.5. DMAMUX mapping ..... 650
**18.6. Register definition ..... 659**
18.6.1. Request multiplexer channel x configuration register (DMAMUX_RM_CHxCFG) ..... 659
18.6.2. Request multiplexer channel interrupt flag register (DMAMUX_RM_INTF) ..... 660
18.6.3. Request multiplexer channel interrupt flag clear register (DMAMUX_RM_INTC) ..... 660
18.6.4. Request generator channel x configuration register (DMAMUX_RG_CHxCFG) ..... 661
18.6.5. Request generator channel interrupt flag register (DMAMUX_RG_INTF) ..... 662
18.6.6. Rquest generator channel interrupt flag clear register (DMAMUX_RG_INTC) ..... 662
**19. Debug (DBG) ..... 664**
**19.1. Overview ..... 664**
**19.2. JTAG / SW function overview ..... 664**
19.2.1. Switch JTAG or SW interface ..... 664
19.2.2. Pin assignment ..... 664
19.2.3. JTAG ..... 665
19.2.4. Debug reset ..... 667
19.2.5. JEDEC-106 ID code ..... 667
**19.3. Debug hold function overview ..... 667**
19.3.1. Debug support for power saving mode ..... 667
19.3.2. Debug support for TIMER, I2C, WWDGT, FWDGT, RTC and CAN ..... 667
**19.4. Register definition ..... 668**
19.4.1. ID code register (DBG_ID) ..... 668
19.4.2. Control register0 (DBG_CTL0) ..... 668
19.4.3. Control register1 (DBG_CTL1) ..... 669
19.4.4. Control register2 (DBG_CTL2) ..... 669
19.4.5. Control register3 (DBG_CTL3) ..... 672

19.4.6. Control register4 (DBG_CTL4) ..... 673
20. Analog-to-digital converter (ADC) ..... 675
20.1. Overview ..... 675
20.2. Characteristics ..... 675
20.3. Pins and internal signals ..... 676
20.4. Function overview ..... 677
20.4.1. Foreground calibration function ..... 677
20.4.2. Dual clock domain architecture ..... 678
20.4.3. ADC enable ..... 678
20.4.4. Single-enden and differential input channels ..... 678
20.4.5. Routine sequence ..... 679
20.4.6. Operation modes ..... 680
20.4.7. Conversion result threshold monitor function ..... 682
20.4.8. Data storage mode ..... 683
20.4.9. Sample time configuration ..... 684
20.4.10. External trigger configuration ..... 684
20.4.11. DMA request ..... 684
20.4.12. Overflow detection ..... 684
20.4.13. ADC internal channels ..... 685
20.4.14. Battery voltage monitoring ..... 686
20.4.15. Using HPDF to managing the conversion results ..... 687
20.4.16. Programmable resolution (DRES) ..... 687
20.4.17. On-chip hardware oversampling ..... 688
20.5. ADC sync mode ..... 690
20.5.1. Free mode ..... 691
20.5.2. Routine parallel mode ..... 691
20.5.3. Routine follow-up mode ..... 692
20.5.4. Use DMA in ADC sync mode ..... 693
20.6. ADC interrupts ..... 694
20.7. Register definition ..... 695
20.7.1. Status register (ADC_STAT) ..... 695
20.7.2. Control register 0 (ADC_CTL0) ..... 696
20.7.3. Control register 1 (ADC_CTL1) ..... 698
20.7.4. Watchdog high threshold register0 (ADC_WDHT0) ..... 700
20.7.5. Watchdog low threshold register0 (ADC_WDLT0) ..... 701
20.7.6. Routine sequence register 0 (ADC_RSQ0) ..... 701
20.7.7. Routine sequence register 1 (ADC_RSQ1) ..... 702
20.7.8. Routine sequence register 2 (ADC_RSQ2) ..... 703
20.7.9. Routine sequence register 3 (ADC_RSQ3) ..... 704
20.7.10. Routine sequence register 4 (ADC_RSQ4) ..... 705
20.7.11. Routine sequence register 5 (ADC_RSQ5) ..... 706

20.7.12. Routine sequence register 6 (ADC_RSQ6) ..... 707
20.7.13. Routine sequence register 7 (ADC_RSQ7) ..... 708
20.7.14. Routine sequence register 8 (ADC_RSQ8) ..... 709
20.7.15. Routine data register (ADC_RDATA) ..... 710
20.7.16. Oversample control register (ADC_OVSAMPCTL) ..... 710
20.7.17. Watchdog 1 Channel Selection Register (ADC_WD1SR) ..... 711
20.7.18. Watchdog 2 Channel Selection Register (ADC_WD2SR) ..... 712
20.7.19. Watchdog high threshold register1 (ADC_WDHT1) ..... 713
20.7.20. Watchdog low threshold register1 (ADC_WDLT1) ..... 713
20.7.21. Watchdog high threshold register2 (ADC_WDHT2) ..... 714
20.7.22. Watchdog low threshold register2 (ADC_WDLT2) ..... 714
20.7.23. Differential mode control register (ADC_DIFCTL) ..... 715
20.7.24. Summary status register (ADC_SSTAT) ..... 715
20.7.25. Sync control register (ADC_SYNCCTL) ..... 716
20.7.26. Sync routine data register0 (ADC_SYNCDATA0) ..... 718
20.7.27. Sync routine data register1 (ADC_SYNCDATA1) ..... 719

**21. Digital-to-analog converter (DAC) ..... 720**

**21.1. Overview ..... 720**

**21.2. Characteristics ..... 720**

**21.3. Function description ..... 722**

21.3.1. DAC enable ..... 722
21.3.2. DAC output buffer ..... 722
21.3.3. DAC data configuration ..... 722
21.3.4. DAC trigger ..... 722
21.3.5. DAC conversion ..... 722
21.3.6. DAC noise wave ..... 723
21.3.7. DAC output voltage ..... 724
21.3.8. DMA request ..... 724
21.3.9. DAC concurrent conversion ..... 724
21.3.10. DAC output buffer calibration ..... 725
21.3.11. DAC modes ..... 725
21.3.12. DAC low-power modes ..... 727

**21.4. DAC register ..... 728**

21.4.1. DACx control register 0 (DAC_CTL0) ..... 728
21.4.2. DACx software trigger register (DAC_SWT) ..... 730
21.4.3. DACx_OUT0 12-bit right-aligned data holding register (DAC_OUT0_R12DH) ..... 731
21.4.4. DACx_OUT0 12-bit left-aligned data holding register (DAC_OUT0_L12DH) ..... 731
21.4.5. DACx_OUT0 8-bit right-aligned data holding register (DAC_OUT0_R8DH) ..... 732
21.4.6. DACx_OUT1 12-bit right-aligned data holding register (DAC_OUT1_R12DH) ..... 732
21.4.7. DACx_OUT1 12-bit left-aligned data holding register (DAC_OUT1_L12DH) ..... 733
21.4.8. DACx_OUT1 8-bit right-aligned data holding register (DAC_OUT1_R8DH) ..... 733
21.4.9. DACx concurrent mode 12-bit right-aligned data holding register (DACC_R12DH) ..... 734

21.4.10. DACx concurrent mode 12-bit left-aligned data holding register (DACC_L12DH) ..... 734
21.4.11. DACx concurrent mode 8-bit right-aligned data holding register (DACC_R8DH) ..... 735
21.4.12. DACx_OUT0 data output register (DAC_OUT0_DO) ..... 735
21.4.13. DACx_OUT1 data output register (DAC_OUT1_DO) ..... 736
21.4.14. DACx status register 0 (DAC_STAT0) ..... 736
21.4.15. DACx calibration register (DAC_CALR) ..... 737
21.4.16. DACx mode control register (DAC_MDCR) ..... 738
21.4.17. DACx sample and keep sample time register 0 (DAC_SKSTR0) ..... 739
21.4.18. DACx sample and keep sample time register 1 (DAC_SKSTR1) ..... 739
21.4.19. DACx sample and keep keep time register (DAC_SKKTR) ..... 740
21.4.20. DACx sample and keep refresh time register (DAC_SKRTR) ..... 740

**22. Watchdog timer (WDGT) ..... 741**

**22.1. Free watchdog timer (FWDGT) ..... 741**
22.1.1. Overview ..... 741
22.1.2. Characteristics ..... 741
22.1.3. Function overview ..... 741
22.1.4. Register definition ..... 744

**22.2. Window watchdog timer (WWDGT) ..... 748**
22.2.1. Overview ..... 748
22.2.2. Characteristics ..... 748
22.2.3. Function overview ..... 748
22.2.4. Register definition ..... 751

**23. Real time clock (RTC) ..... 753**

**23.1. Overview ..... 753**

**23.2. Characteristics ..... 753**

**23.3. Function overview ..... 754**
23.3.1. Block diagram ..... 754
23.3.2. Clock source and prescalers ..... 754
23.3.3. Shadow registers introduction ..... 755
23.3.4. Configurable and field maskable alarm ..... 755
23.3.5. Configurable periodic auto-wakeup counter ..... 756
23.3.6. RTC initialization and configuration ..... 756
23.3.7. Calendar reading ..... 757
23.3.8. Resetting the RTC ..... 759
23.3.9. RTC shift function ..... 759
23.3.10. RTC reference clock detection ..... 760
23.3.11. RTC smooth digital calibration ..... 760
23.3.12. Time-stamp function ..... 762
23.3.13. Tamper detection ..... 762
23.3.14. Calibration clock output ..... 764
23.3.15. Alarm output ..... 764
23.3.16. RTC pin configuration ..... 764

23.3.17. RTC power saving mode management ..... 765
23.3.18. RTC interrupts ..... 766
**23.4. Register definition ..... 767**
23.4.1. Time register (RTC_TIME) ..... 767
23.4.2. Date register (RTC_DATE) ..... 767
23.4.3. Control register (RTC_CTL) ..... 768
23.4.4. Status register (RTC_STAT) ..... 770
23.4.5. Prescaler register (RTC_PSC) ..... 773
23.4.6. Wakeup timer register (RTC_WUT) ..... 773
23.4.7. Alarm 0 time and date register (RTC_ALRM0TD) ..... 774
23.4.8. Alarm 1 time and date register (RTC_ALRM1TD) ..... 775
23.4.9. Write protection key register (RTC_WPK) ..... 776
23.4.10. Sub second register (RTC_SS) ..... 776
23.4.11. Shift function control register (RTC_SHIFTCTL) ..... 777
23.4.12. Time of time stamp register (RTC_TTS) ..... 777
23.4.13. Date of time stamp register (RTC_DTS) ..... 778
23.4.14. Sub second of time stamp register (RTC_SSTS) ..... 779
23.4.15. High resolution frequency compensation register (RTC_HRFC) ..... 779
23.4.16. Tamper register (RTC_TAMP) ..... 780
23.4.17. Alarm 0 sub second register (RTC_ALRM0SS) ..... 782
23.4.18. Alarm 1 sub second register (RTC_ALRM1SS) ..... 783
23.4.19. Configuration register (RTC_CFG) ..... 784
23.4.20. Backup registers (RTC_BKPx) (x=0..31) ..... 784
**24. TIMER (TIMERx) ..... 786**
**24.1. Advanced timer (TIMERx, x=0, 7) ..... 789**
24.1.1. Overview ..... 789
24.1.2. Characteristics ..... 789
24.1.3. Block diagram ..... 790
24.1.4. Function overview ..... 790
24.1.5. Registers definition (TIMERx, x=0, 7) ..... 832
**24.2. General level0 timer (TIMERx, x=1,2,3,4,22,23,30,31) ..... 900**
24.2.1. Overview ..... 900
24.2.2. Characteristics ..... 900
24.2.3. Block diagram ..... 900
24.2.4. Function overview ..... 901
24.2.5. Registers definition (TIMERx, x=1,2,3,4,22,23,30,31) ..... 926
**24.3. General level3 timer (TIMERx, x=14,40,41,42,43,44) ..... 957**
24.3.1. Overview ..... 957
24.3.2. Characteristics ..... 957
24.3.3. Block diagram ..... 957
24.3.4. Function overview ..... 958
24.3.5. Register definition (TIMERx, x=14,40,41,42,43,44) ..... 984

**24.4. General level4 timer (TIMERx, x=15,16) ..... 1017**
24.4.1. Overview ..... 1017
24.4.2. Characteristics ..... 1017
24.4.3. Block diagram ..... 1017
24.4.4. Function overview ..... 1018
24.4.5. Register definition (TIMERx, x=15,16) ..... 1035
**24.5. Basic timer (TIMERx, x=5,6,50,51) ..... 1061**
24.5.1. Overview ..... 1061
24.5.2. Characteristics ..... 1061
24.5.3. Block diagram ..... 1061
24.5.4. Function overview ..... 1061
24.5.5. Registers definition (TIMERx, x=5,6,50,51) ..... 1065
**25. Universal synchronous / asynchronous receiver / transmitter (USART) ... 1072**
**25.1. Overview ..... 1072**
**25.2. Characteristics ..... 1072**
**25.3. Function overview ..... 1074**
25.3.1. USART frame format ..... 1074
25.3.2. Baud rate generation ..... 1075
25.3.3. USART transmitter ..... 1076
25.3.4. USART receiver ..... 1077
25.3.5. Use DMA for data buffer access ..... 1078
25.3.6. Hardware flow control ..... 1080
25.3.7. Multi-processor communication ..... 1081
25.3.8. LIN mode ..... 1082
25.3.9. Synchronous mode ..... 1083
25.3.10. IrDA SIR ENDEC mode ..... 1084
25.3.11. Half-duplex communication mode ..... 1086
25.3.12. Smartcard (ISO7816-3) mode ..... 1086
25.3.13. ModBus communication ..... 1088
25.3.14. Receive / Transmitter FIFO ..... 1088
25.3.15. Wakeup from deep-sleep mode ..... 1089
25.3.16. USART interrupts ..... 1090
**25.4. Register definition ..... 1092**
25.4.1. Control register 0 (USART_CTL0) ..... 1092
25.4.2. Control register 1 (USART_CTL1) ..... 1095
25.4.3. Control register 2 (USART_CTL2) ..... 1097
25.4.4. Baud rate generator register (USART_BAUD) ..... 1101
25.4.5. Prescaler and guard time configuration register (USART_GP) ..... 1101
25.4.6. Receiver timeout register (USART_RT) ..... 1102
25.4.7. Command register (USART_CMD) ..... 1103
25.4.8. Status register (USART_STAT) ..... 1104
25.4.9. Interrupt status clear register (USART_INTC) ..... 1108

25.4.10. Receive data register (USART_RDATA) ..... 1110
25.4.11. Transmit data register (USART_TDATA) ..... 1110
25.4.12. USART coherence control register (USART_CHC) ..... 1111
25.4.13. USART FIFO control and status register (USART_FCS) ..... 1111
**26. Inter-integrated circuit interface (I2C) ..... 1115**
**26.1. Overview ..... 1115**
**26.2. Characteristics ..... 1115**
**26.3. Function overview ..... 1115**
26.3.1. Clock requirements ..... 1116
26.3.2. I2C communication flow ..... 1117
26.3.3. Noise filter ..... 1120
26.3.4. I2C timings configuration ..... 1120
26.3.5. I2C reset ..... 1122
26.3.6. Data transfer ..... 1122
26.3.7. I2C slave mode ..... 1124
26.3.8. I2C master mode ..... 1129
26.3.9. SMBus support ..... 1134
26.3.10. SMBus mode ..... 1137
26.3.11. Wakeup from power saving mode ..... 1138
26.3.12. Use DMA for data transfer ..... 1139
26.3.13. I2C error and interrupts ..... 1139
26.3.14. I2C debug mode ..... 1140
**26.4. Register definition ..... 1141**
26.4.1. Control register 0 (I2C_CTL0) ..... 1141
26.4.2. Control register 1 (I2C_CTL1) ..... 1143
26.4.3. Slave address register 0 (I2C_SADDR0) ..... 1145
26.4.4. Slave address register 1 (I2C_SADDR1) ..... 1146
26.4.5. Timing register (I2C_TIMING) ..... 1147
26.4.6. Timeout register (I2C_TIMEOUT) ..... 1148
26.4.7. Status register (I2C_STAT) ..... 1149
26.4.8. Status clear register (I2C_STATC) ..... 1152
26.4.9. PEC register (I2C_PEC) ..... 1153
26.4.10. Receive data register (I2C_RDATA) ..... 1153
26.4.11. Transmit data register (I2C_TDATA) ..... 1154
26.4.12. Control register 2 (I2C_CTL2) ..... 1154
**27. Serial peripheral interface/Inter-IC sound (SPI/I2S) ..... 1155**
**27.1. Overview ..... 1155**
**27.2. Characteristics ..... 1155**
27.2.1. SPI characteristics ..... 1155
27.2.2. I2S characteristics ..... 1155
**27.3. SPI function overview ..... 1156**

27.3.1. SPI block diagram ..... 1156
27.3.2. SPI signal description ..... 1157
27.3.3. SPI clock timing and data format ..... 1158
27.3.4. SPI clock delay mode ..... 1159
27.3.5. RxFIFO and TxFIFO ..... 1161
27.3.6. NSS function ..... 1164
27.3.7. SPI operation modes ..... 1166
27.3.8. DMA function ..... 1175
27.3.9. CRC function ..... 1175
27.3.10. SPI interrupts ..... 1176
**27.4. I2S function overview ..... 1180**
27.4.1. I2S block diagram ..... 1180
27.4.2. I2S signal description ..... 1180
27.4.3. I2S audio standards ..... 1181
27.4.4. I2S clock ..... 1188
27.4.5. RxFIFO and TxFIFO ..... 1189
27.4.6. Operation ..... 1190
27.4.7. DMA function ..... 1192
27.4.8. I2S interrupts ..... 1192
**27.5. Register definition ..... 1195**
27.5.1. Control register 0 (SPI_CTL0) ..... 1195
27.5.2. Control register 1 (SPI_CTL1) ..... 1197
27.5.3. Configuration register 0 (SPI_CFG0) ..... 1197
27.5.4. Configuration register 1 (SPI_CFG1) ..... 1200
27.5.5. Interrupt register (SPI_INT) ..... 1202
27.5.6. Status register (SPI_STAT) ..... 1204
27.5.7. Interrupt/Status flags clear register (SPI_STATC) ..... 1206
27.5.8. Data Transfer register (SPI_TDATA) ..... 1207
27.5.9. Data Receive register (SPI_RDATA) ..... 1207
27.5.10. CRC polynomial register (SPI_CRCPOLY) ..... 1208
27.5.11. TX CRC register (SPI_TCRC) ..... 1208
27.5.12. RX CRC register (SPI_RCRC) ..... 1209
27.5.13. Underrun Data register (SPI_URDATA) ..... 1209
27.5.14. I2S control register (SPI_I2SCTL) ..... 1210
27.5.15. Quad_SPI mode control register (SPI_QCTL) ..... 1212
27.5.16. RX clock delay register (SPI_RXDLYCK) ..... 1212
**28. OSPI I/O manager(OSPIM) ..... 1214**
**28.1. Overview ..... 1214**
**28.2. Characteristics ..... 1214**
**28.3. Function overview ..... 1214**
28.3.1. OSPIM block diagram ..... 1214
28.3.2. OSPIM matrix ..... 1215

**28.4. Register definition ..... 1215**
28.4.1. Port configuration register (OSPIM_PCFGx) (x = 0, 1) ..... 1215
**29. Octal-SPI interface(OSPI) ..... 1217**
**29.1. Overview ..... 1217**
**29.2. Characteristics ..... 1217**
**29.3. Functon overview ..... 1217**
29.3.1. OSPI block diagram ..... 1217
29.3.2. OSPI regular command format ..... 1220
**29.4. Operating modes ..... 1223**
29.4.1. Indirect mode ..... 1223
29.4.2. Status polling mode ..... 1224
29.4.3. Memory map mode ..... 1224
**29.5. OSPI configuration ..... 1225**
29.5.1. OSPI system configuration ..... 1225
29.5.2. OSPI device configuration ..... 1225
29.5.3. OSPI regular commamd configuration ..... 1225
**29.6. Data sampling shift ..... 1226**
**29.7. Busy ..... 1226**
**29.8. Error management ..... 1226**
**29.9. OSPI interrupt ..... 1227**
**29.10. Register definition ..... 1228**
29.10.1. Control register (OSPI_CTL) ..... 1228
29.10.2. Device configuration register 0 (OSPI_DCFG0) ..... 1229
29.10.3. Device configuration register 1 (OSPI_DFCG1) ..... 1231
29.10.4. Status register (OSPI_STAT) ..... 1231
29.10.5. Status clear register (OSPI_STATC) ..... 1232
29.10.6. Data length register (OSPI_DTLEN) ..... 1233
29.10.7. Address register(OSPI_ADDR) ..... 1234
29.10.8. Data register (OSPI_DATA) ..... 1234
29.10.9. Status mask register (OSPI_STATMK) ..... 1235
29.10.10. Status match register(OSPI_STATMATCH) ..... 1235
29.10.11. Interval register (OSPI_INTERVAL) ..... 1236
29.10.12. Transfer configuration register (OSPI_TCFG) ..... 1236
29.10.13. Timing configuration register (OSPI_TIMCFG) ..... 1238
29.10.14. Instruction register (OSPI_INS) ..... 1239
29.10.15. Alternate bytes register (OSPI_ALTE) ..... 1239
29.10.16. Wrap transfer configuration register (OSPI_WPTCFG) ..... 1240
29.10.17. Wrap timing configuration register (OSPI_WPTIMCFG) ..... 1242
29.10.18. Wrap instruction register (OSPI_WPINS) ..... 1243
29.10.19. Wrap alternate byte register (OSPI_WPALTE) ..... 1243

29.10.20. Write transfer configuration register (OSPI_WTCFG) ..... 1244
29.10.21. Write timing configuration register (OSPI_WTIMCFG) ..... 1246
29.10.22. Write instruction register (OSPI_WINS) ..... 1246
29.10.23. Write alternate byte register (OSPI_WALTE) ..... 1247
**30. Clock phase delay module (CPDM) (For GD32H737_757_759) ..... 1248**
**30.1. Overview ..... 1248**
**30.2. Characteristics ..... 1248**
**30.3. Function overview ..... 1248**
30.3.1. Overview ..... 1249
30.3.2. Operation process ..... 1250
**30.4. Register definition ..... 1252**
30.4.1. Control register (CPDM_CTL) ..... 1252
30.4.2. Configuration register (CPDM_CFG) ..... 1252
**31. Digital camera interface (DCI) (For GD32H737_757_759) ..... 1254**
**31.1. Overview ..... 1254**
**31.2. Characteristics ..... 1254**
**31.3. Block diagram ..... 1254**
**31.4. Signal description ..... 1255**
**31.5. Function overview ..... 1255**
31.5.1. DCI hardware synchronization mode ..... 1255
31.5.2. Embedded synchronization mode ..... 1256
31.5.3. CCIR656 mode ..... 1256
31.5.4. Capture data using snapshot or continuous capture modes ..... 1259
31.5.5. Window function ..... 1259
31.5.6. Pixel formats, data padding and DMA ..... 1259
**31.6. Interrupts ..... 1260**
**31.7. Register definition ..... 1261**
31.7.1. Control register (DCI_CTL) ..... 1261
31.7.2. Status register0 (DCI_STAT0) ..... 1262
31.7.3. Status register1 (DCI_STAT1) ..... 1263
31.7.4. Interrupt enable register (DCI_INTEN) ..... 1264
31.7.5. Interrupt flag register (DCI_INTF) ..... 1265
31.7.6. Interrupt flag clear register (DCI_INTC) ..... 1266
31.7.7. Synchronization codes register (DCI_SC) ..... 1267
31.7.8. Synchronization codes unmask register (DCI_SCUMSK) ..... 1267
31.7.9. Cropping window start position register (DCI_CWSPOS) ..... 1268
31.7.10. Cropping window size register (DCI_CWSZ) ..... 1268
31.7.11. DATA register (DCI_DATA) ..... 1269
**32. TFT-LCD interface (TLI) (For GD32H737_757_759) ..... 1270**

**32.1. Overview ..... 1270**
**32.2. Characteristics ..... 1270**
**32.3. Block diagram ..... 1270**
**32.4. Signal description ..... 1271**
**32.5. Function overview ..... 1271**
32.5.1. LCD display timing ..... 1271
32.5.2. Pixel DMA function ..... 1272
32.5.3. Pixel formats ..... 1273
32.5.4. Layer window and blending function ..... 1273
32.5.5. Layer configuration reload ..... 1274
32.5.6. Dithering function ..... 1275
**32.6. Interrupts ..... 1275**
**32.7. Register definition ..... 1276**
32.7.1. Synchronous pulse size register (TLI_SPSZ) ..... 1276
32.7.2. Back-porch size register (TLI_BPSZ) ..... 1276
32.7.3. Active size register (TLI_ASZ) ..... 1277
32.7.4. Total size register (TLI_TSZ) ..... 1277
32.7.5. Control register (TLI_CTL) ..... 1278
32.7.6. Reload layer register (TLI_RL) ..... 1279
32.7.7. Background color register (TLI_BGC) ..... 1280
32.7.8. Interrupt enable register (TLI_INTEN) ..... 1280
32.7.9. Interrupt flag register (TLI_INTF) ..... 1281
32.7.10. Interrupt flag clear register (TLI_INTC) ..... 1281
32.7.11. Line mark register (TLI_LM) ..... 1282
32.7.12. Current pixel position register (TLI_CPPOS) ..... 1282
32.7.13. Status register (TLI_STAT) ..... 1283
32.7.14. Layer x control register (TLI_LxCTL) (x = 0,1) ..... 1283
32.7.15. Layer x horizontal position parameters register (TLI_LxHPOS) (x = 0,1) ..... 1284
32.7.16. Layer x vertical position parameters register (TLI_LxVPOS) (x = 0,1) ..... 1284
32.7.17. Layer x color key register (TLI_LxCKEY) (x = 0,1) ..... 1285
32.7.18. Layer x packeted pixel format register (TLI_LxPPF) (x = 0,1) ..... 1285
32.7.19. Layer x specified alpha register (TLI_LxSA) (x = 0,1) ..... 1286
32.7.20. Layer x default color register (TLI_LxDC) (x = 0,1) ..... 1286
32.7.21. Layer x blending register (TLI_LxBLEND) (x = 0,1) ..... 1287
32.7.22. Layer x frame base address register (TLI_LxFBADDR) (x = 0,1) ..... 1288
32.7.23. Layer x frame line length register (TLI_LxFLLEN) (x = 0,1) ..... 1288
32.7.24. Layer x frame total line number register (TLI_LxFTLN) (x = 0,1) ..... 1289
32.7.25. Layer x look up table register (TLI_LxLUT) (x = 0,1) ..... 1289
**33. Receiver of Sony/Philips Digital Interface (RSPDIF) (For GD32H737_757_759) 1290**
**33.1. Overview ..... 1290**

33.2. Characteristics ..... 1290
33.3. Function overview ..... 1290
33.3.1. RSPDIF block diagram ..... 1290
33.3.2. S/PDIF protocol ..... 1292
33.3.3. RSPDIF ..... 1294
33.3.4. RSPDIF synchronization process ..... 1298
33.3.5. RSPDIF state machine ..... 1300
33.3.6. RSPDIF data reception management ..... 1303
33.3.7. RSPDIF clock management ..... 1306
33.3.8. DMA function ..... 1307
33.3.9. Status、error and interrupt ..... 1308
33.4. Register definition ..... 1313
33.4.1. Control register (RSPDIF_CTL) ..... 1313
33.4.2. Interrupt enable register (RSPDIF_INTEN) ..... 1315
33.4.3. Status register (RSPDIF_STAT) ..... 1316
33.4.4. Status flag clear register (RSPDIF_STATC) ..... 1319
33.4.5. RX data register (RSPDIF_DATA) ..... 1319
33.4.6. RX Channel status register (RSPDIF_CHSTAT) ..... 1321
33.4.7. RX data threshold register (RSPDIF_DTH) ..... 1322
34. Serial Audio Interface (SAI) (For GD32H737_757_759) ..... 1324
34.1. Overview ..... 1324
34.2. Characteristics ..... 1324
34.3. Function overview ..... 1325
34.3.1. Block diagram ..... 1325
34.3.2. Clock divider ..... 1326
34.3.3. Operating mode ..... 1327
34.3.4. SAI synchronization mode ..... 1328
34.3.5. Frame configuration ..... 1329
34.3.6. Slot configuration ..... 1331
34.3.7. Data configuration ..... 1333
34.3.8. Internal FIFO ..... 1334
34.3.9. PDM interface ..... 1335
34.3.10. AC’97 link controller ..... 1341
34.3.11. SPDIF Output ..... 1343
34.3.12. Stereo/Mono ..... 1344
34.3.13. Mute ..... 1344
34.3.14. Compander ..... 1345
34.3.15. Output drive ..... 1348
34.3.16. IO management ..... 1348
34.3.17. DMA interface ..... 1348
34.3.18. Enable/Disable ..... 1349
34.3.19. Error flags ..... 1349

34.3.20. Interrupts ..... 1352
**34.4. Register definition ..... 1353**
34.4.1. Synchronize configuration register (SAI_SYNCFG) ..... 1353
34.4.2. Block x configuration register0 (SAI_BxCFG0) (x = 0,1) ..... 1353
34.4.3. Block x configuration register1 (SAI_BxCFG1) (x = 0,1) ..... 1356
34.4.4. Block x frame configuration register (SAI_BxFCFG) (x = 0,1) ..... 1358
34.4.5. Block x slot configuration register (SAI_BxSCFG) (x = 0,1) ..... 1359
34.4.6. Block x interrupt enable register (SAI_BxINTEN) (x = 0,1) ..... 1360
34.4.7. Block x status register (SAI_BxSTAT) (x = 0,1) ..... 1362
34.4.8. Block x interrupt flag clear register (SAI_BxINTC) (x = 0,1) ..... 1364
34.4.9. Block x data register (SAI_BxDATA) (x = 0,1) ..... 1365
34.4.10. PDM control register (SAI_PDMCTL) ..... 1365
34.4.11. PDM configuration register (SAI_PDMCFG) ..... 1366
**35. Image processing accelerator (IPA) (For GD32H737_757_759) ..... 1369**
**35.1. Overview ..... 1369**
**35.2. Characteristics ..... 1369**
**35.3. Block diagram ..... 1370**
**35.4. Signal description ..... 1370**
**35.5. Function overview ..... 1371**
35.5.1. Conversion operation ..... 1372
35.5.2. Foreground and background LUT ..... 1372
35.5.3. Foreground and background pixel channel extension (PCE) ..... 1373
35.5.4. Foreground channel scaling ..... 1377
35.5.5. Blending ..... 1378
35.5.6. Destination pixel channel compression (PCC) ..... 1379
35.5.7. Rotation ..... 1380
35.5.8. Inter-timer ..... 1380
35.5.9. Line mark ..... 1381
35.5.10. Transfer flow ..... 1381
35.5.11. Configuration ..... 1382
**35.6. Interrupts ..... 1386**
**35.7. Register definition ..... 1389**
35.7.1. Control register (IPA_CTL) ..... 1389
35.7.2. Interrupt flag register (IPA_INTF) ..... 1390
35.7.3. Interrupt flag clear register (IPA_INTC) ..... 1391
35.7.4. Foreground memory base address register (IPA_FMADDR) ..... 1392
35.7.5. Foreground line offset register (IPA_FLOFF) ..... 1393
35.7.6. Background memory base address register (IPA_BMADDR) ..... 1393
35.7.7. Background line offset register (IPA_BLOFF) ..... 1394
35.7.8. Foreground pixel control register (IPA_FPCTL) ..... 1394
35.7.9. Foreground pixel value register (IPA_FPV) ..... 1396

35.7.10. Background pixel control register (IPA_BPCTL) ..... 1397
35.7.11. Background pixel value register (IPA_BPV) ..... 1398
35.7.12. Foreground LUT memory base address register (IPA_FLMADDR) ..... 1399
35.7.13. Background LUT memory base address register (IPA_BLMADDR) ..... 1400
35.7.14. Destination pixel control register (IPA_DPCTL) ..... 1400
35.7.15. Destination pixel value register (IPA_DPV) ..... 1401
35.7.16. Destination memory base address register (IPA_DMADDR) ..... 1404
35.7.17. Destination line offset register (IPA_DLOFF) ..... 1405
35.7.18. Image size register (IPA_IMS) ..... 1406
35.7.19. Line mark register (IPA_LM) ..... 1406
35.7.20. Inter-timer control register (IPA_ITCTL) ..... 1407
35.7.21. Bilinear scaling control register (IPA_BSCTL) ..... 1407
35.7.22. Scaling destination image size register (IPA_DIMS) ..... 1408
35.7.23. Foreground even frame/UV memory base address register (IPA_EF_UV_MADDR) ..... 1409
35.7.24. Color space conversion coefficient configure register 0 (IPA_CSCC_CFG0) ..... 1409
35.7.25. Color space conversion coefficient configure register 1 (IPA_CSCC_CFG1) ..... 1410
35.7.26. Color space conversion coefficient configure register 2 (IPA_CSCC_CFG2) ..... 1410

**36. Secure digital input/output interface (SDIO) (For GD32H737_757_759) ..... 1412**

**36.1. Overview ..... 1412**

**36.2. Characteristics ..... 1412**

**36.3. SDIO function overview ..... 1413**

36.3.1. SDIO bus topology ..... 1413
36.3.2. SDIO operation modes ..... 1415
36.3.3. Block diagram ..... 1416
36.3.4. SDIO pins and internal signals ..... 1416
36.3.5. General description ..... 1417
36.3.6. SDIO adapter ..... 1418
36.3.7. AHB slave interface ..... 1423
36.3.8. AHB master interface ..... 1424
36.3.9. MDMA request ..... 1425
36.3.10. AHB and SDIO_CLK clock relation ..... 1426
36.3.11. Hardware flow control ..... 1426

**36.4. Card function overview ..... 1427**

36.4.1. Card registers ..... 1427
36.4.2. Commands ..... 1428
36.4.3. Responses ..... 1440
36.4.4. Data packets format ..... 1444
36.4.5. Two status fields of the card ..... 1446

**36.5. Programming sequence ..... 1453**

36.5.1. Card identification ..... 1453
36.5.2. Boot operation ..... 1455
36.5.3. No data commands ..... 1457

36.5.4. Single block or multiple block write ..... 1458
36.5.5. Single block or multiple block read ..... 1459
36.5.6. Stream write and stream read (e•MMC only) ..... 1460
36.5.7. Erase ..... 1461
36.5.8. Bus width selection ..... 1462
36.5.9. Protection management ..... 1462
36.5.10. Card Lock/Unlock operation ..... 1463
36.5.11. Sleep ..... 1465
36.5.12. CMD12 send timing ..... 1466
**36.6. Specific operations ..... 1469**
36.6.1. UHS-I ..... 1469
**36.7. Register definition ..... 1471**
36.7.1. Power control register (SDIO_PWRCTL) ..... 1471
36.7.2. Clock control register (SDIO_CLKCTL) ..... 1472
36.7.3. Command argument register (SDIO_CMDAGMT) ..... 1474
36.7.4. Command control register (SDIO_CMDCTL) ..... 1474
36.7.5. Command index response register (SDIO_RSPCMDIDX) ..... 1476
36.7.6. Response register (SDIO_RESPx x = 0..3) ..... 1477
36.7.7. Data timeout register (SDIO_DATATO) ..... 1477
36.7.8. Data length register (SDIO_DATALEN) ..... 1478
36.7.9. Data control register (SDIO_DATACTL) ..... 1478
36.7.10. Data counter register (SDIO_DATACNT) ..... 1480
36.7.11. Status register (SDIO_STAT) ..... 1481
36.7.12. Interrupt clear register (SDIO_INTC) ..... 1482
36.7.13. Interrupt enable register (SDIO_INTEN) ..... 1484
36.7.14. ACK timeout register (SDIO_ACKTO) ..... 1486
36.7.15. FIFO data register (SDIO_FIFO) ..... 1486
36.7.16. IDMA control register (SDIO_IDMACTL) ..... 1487
36.7.17. IDMA buffer size register (SDIO_IDMASIZE) ..... 1487
36.7.18. IDMA buffer 0 base address register (SDIO_IDMAADDR0) ..... 1488
36.7.19. IDMA buffer 1 base address register (SDIO_IDMAADDR1) ..... 1488
**37. Management data input / output (MDIO) (For GD32H737_757_759) ..... 1490**
**37.1. Overview ..... 1490**
**37.2. Characteristics ..... 1490**
**37.3. Pins and internal signals ..... 1490**
**37.4. Function overview ..... 1491**
37.4.1. Frame structure ..... 1491
37.4.2. Typical Usage Sequence ..... 1493
**37.5. Register definition ..... 1495**
37.5.1. Control register (MDIO_CTL) ..... 1495
37.5.2. Received frame information register (MDIO_RFRM) ..... 1495

37.5.3. Received data register (MDIO_RDATA) ..... 1496
37.5.4. Received address register (MDIO_RADDR) ..... 1496
37.5.5. Transfer data register (MDIO_TDATA) ..... 1497
37.5.6. Configuration register (MDIO_CFG) ..... 1497
37.5.7. Status register (MDIO_STAT) ..... 1498
37.5.8. Interrupt enable register (MDIO_INTEN) ..... 1499
37.5.9. Pin value register (MDIO_PIN) ..... 1500
37.5.10. Timeout register (MDIO_TO) ..... 1500
**38. External memory controller (EXMC) ..... 1502**
**38.1. Overview ..... 1502**
**38.2. Characteristics ..... 1502**
**38.3. Function overview ..... 1502**
38.3.1. Block diagram ..... 1502
38.3.2. Bus interface ..... 1503
38.3.3. AXI error ..... 1503
38.3.4. Basic regulation of EXMC access ..... 1504
38.3.5. External device address mapping ..... 1505
38.3.6. NOR/PSRAM controller ..... 1509
38.3.7. NAND flash controller ..... 1528
38.3.8. SDRAM controller ..... 1532
**38.4. Register definition ..... 1544**
38.4.1. NOR/PSRAM controller registers ..... 1544
38.4.2. NAND flash controller registers ..... 1549
38.4.3. SDRAM controller registers ..... 1553
**39. VREF ..... 1561**
**39.1. Overview ..... 1561**
**39.2. Characteristics ..... 1561**
**39.3. Function overview ..... 1561**
**39.4. Register definition ..... 1563**
39.4.1. Control and status register (VREF_CS) ..... 1563
39.4.2. Calibration register (VREF_CALIB) ..... 1563
**40. Low power digital temperature sensor (LPDTS) ..... 1565**
**40.1. Overview ..... 1565**
**40.2. Characteristics ..... 1565**
**40.3. Block diagram ..... 1565**
**40.4. Function overview ..... 1566**
40.4.1. LPDTS internal signals ..... 1566
40.4.2. Operating modes ..... 1566
40.4.3. Temperature measurement principles ..... 1566

40.4.4. Sampling time ..... 1567
40.4.5. Trigger input ..... 1568
40.4.6. On-off control and ready flag ..... 1569
40.4.7. LPDTS low-power modes ..... 1569
40.4.8. LPDTS interrupts ..... 1569
**40.5. Register definition ..... 1571**
40.5.1. Configuration register (LPDTS_CFG) ..... 1571
40.5.2. Sensor T0 data register (LPDTS_SDATA) ..... 1572
40.5.3. Ramp data register (LPDTS_RDATA) ..... 1572
40.5.4. Interrupt threshold register (LPDTS_IT) ..... 1573
40.5.5. Temperature data register (LPDTS_DATA) ..... 1573
40.5.6. Temperature sensor status register (LPDTS_STAT) ..... 1574
40.5.7. Interrupt enable register (LPDTS_INTEN) ..... 1575
40.5.8. Interrupt clear flag register (LPDTS_INTC) ..... 1576
40.5.9. Option register (LPDTS_OP) ..... 1577
**41. Encoder Divided-Output controller (EDOUT) ..... 1578**
**41.1. Overview ..... 1578**
**41.2. Characteristics ..... 1578**
**41.3. Function overview ..... 1578**
**41.4. Z-phase output mode ..... 1579**
**41.5. Operation guidance ..... 1580**
41.5.1. EDOUT initialization ..... 1580
41.5.2. EDOUT update processing ..... 1581
41.5.3. EDOUT working example ..... 1581
**41.6. Register definition ..... 1583**
41.6.1. Control register (EDOUT_CTL) ..... 1583
41.6.2. Enable register (EDOUT_ENABLE) ..... 1583
41.6.3. Location register (EDOUT_LOC) ..... 1584
41.6.4. Output counter register (EDOUT_OCNT) ..... 1584
41.6.5. Location counter register (EDOUT_LCNT) ..... 1585
41.6.6. Z-phase configure register (EDOUT_ZCR) ..... 1586
**42. Controller area network (CAN) ..... 1587**
**42.1. Overview ..... 1587**
**42.2. Characteristics ..... 1587**
**42.3. Function overview ..... 1588**
42.3.1. Mailbox descriptor ..... 1589
42.3.2. Rx FIFO descriptor ..... 1594
42.3.3. Communication modes ..... 1599
42.3.4. Power saving modes ..... 1601
42.3.5. Data transmission ..... 1602

42.3.6. Data reception ..... 1606
42.3.7. Data reception in Pretended Networking mode ..... 1613
42.3.8. CAN FD operation ..... 1616
42.3.9. Errors and states ..... 1619
42.3.10. Communication parameters ..... 1621
42.3.11. Interrupts ..... 1625
**42.4. Example for a typical configuration flow of CAN ..... 1625**
**42.5. CAN registers ..... 1628**
42.5.1. Control register 0 (CAN_CTL0) ..... 1628
42.5.2. Control register 1 (CAN_CTL1) ..... 1630
42.5.3. Timer register (CAN_TIMER) ..... 1632
42.5.4. Receive mailbox public filter register (CAN_RMPUBF) ..... 1632
42.5.5. Error register 0 (CAN_ERR0) ..... 1633
42.5.6. Error register 1 (CAN_ERR1) ..... 1633
42.5.7. Interrupt enable register (CAN_INTEN) ..... 1636
42.5.8. Status register (CAN_STAT) ..... 1637
42.5.9. Control register 2 (CAN_CTL2) ..... 1638
42.5.10. CRC for classical frame register (CAN_CRCC) ..... 1640
42.5.11. Receive FIFO public filter register (CAN_RFIFOPUBF) ..... 1641
42.5.12. Receive FIFO identifier filter matching number register (CAN_RFIFOIFMN) ..... 1641
42.5.13. Bit timing register (CAN_BT) ..... 1642
42.5.14. Receive FIFO/mailbox private filter x register (CAN_RFIFOMPFx)(x=0..31) ..... 1643
42.5.15. Pretended Networking mode control register 0 (CAN_PN_CTL0) ..... 1643
42.5.16. Pretended Networking mode timeout register (CAN_PN_TO) ..... 1645
42.5.17. Pretended Networking mode status register (CAN_PN_STAT) ..... 1645
42.5.18. Pretended Networking mode expected identifier 0 register (CAN_PN_EID0) ..... 1646
42.5.19. Pretended Networking mode expected DLC register (CAN_PN_EDLC) ..... 1647
42.5.20. Pretended Networking mode expected data low 0 register (CAN_PN_EDL0) ..... 1647
42.5.21. Pretended Networking mode expected data low 1 register (CAN_PN_EDL1) ..... 1648
42.5.22. Pretended Networking mode identifier filter / expected identifier 1 register (CAN_PN_IFEID1) ..... 1648
42.5.23. Pretended Networking mode data 0 filter / expected data high 0 register (CAN_PN_DF0EDH0) ..... 1649
42.5.24. Pretended Networking mode data 1 filter / expected data high 1 register (CAN_PN_DF1EDH1) ..... 1650
42.5.25. Pretended Networking mode received wakeup mailbox x control status information register (CAN_PN_RWMxCS)(x=0..3) ..... 1651
42.5.26. Pretended Networking mode received wakeup mailbox x identifier register (CAN_PN_RWMxI)(x=0..3) ..... 1652
42.5.27. Pretended Networking mode received wakeup mailbox x data 0 register (CAN_PN_RWMxD0)(x=0..3) ..... 1652
42.5.28. Pretended Networking mode received wakeup mailbox x data 1 register (CAN_PN_RWMxD1)(x=0..3) ..... 1653
42.5.29. FD control register (CAN_FDCTL) ..... 1653

42.5.30. FD bit timing register (CAN_FDBT) ..... 1654
42.5.31. CRC for classical and FD frame register (CAN_CRCCFD) ..... 1655

**43. Ethernet (ENET) (For GD32H737_757_759) ..... 1657**

**43.1. Overview ..... 1657**

**43.2. Characteristics ..... 1657**

43.2.1. Block diagram ..... 1658
43.2.2. MAC 802.3 Ethernet packet description ..... 1660
43.2.3. Ethernet signal description ..... 1660

**43.3. Function overview ..... 1662**

43.3.1. Interface configuration ..... 1662
43.3.2. MAC function overview ..... 1666
43.3.3. DMA controller description ..... 1677
43.3.4. MAC statistics counters: MSC ..... 1701
43.3.5. Wake up management: WUM ..... 1702
43.3.6. Precision time protocol: PTP ..... 1705
43.3.7. Example for a typical configuration flow of Ethernet ..... 1710
43.3.8. Ethernet interrupts ..... 1711

**43.4. Register definition ..... 1714**

43.4.1. MAC configuration register (ENET_MAC_CFG) ..... 1714
43.4.2. MAC frame filter register (ENET_MAC_FRMF) ..... 1716
43.4.3. MAC hash list high register (ENET_MAC_HLH) ..... 1718
43.4.4. MAC hash list low register (ENET_MAC_HLL) ..... 1718
43.4.5. MAC PHY control register (ENET_MAC_PHY_CTL) ..... 1719
43.4.6. MAC PHY data register (ENET_MAC_PHY_DATA) ..... 1720
43.4.7. MAC flow control register (ENET_MAC_FCTL) ..... 1720
43.4.8. MAC VLAN tag register (ENET_MAC_VLT) ..... 1722
43.4.9. MAC remote wakeup frame filter register (ENET_MAC_RWFF) ..... 1723
43.4.10. MAC wakeup management register (ENET_MAC_WUM) ..... 1723
43.4.11. MAC debug register (ENET_MAC_DBG) ..... 1724
43.4.12. MAC interrupt flag register (ENET_MAC_INTF) ..... 1726
43.4.13. MAC interrupt mask register (ENET_MAC_INTMSK) ..... 1727
43.4.14. MAC address 0 high register (ENET_MAC_ADDR0H) ..... 1728
43.4.15. MAC address 0 low register (ENET_MAC_ADDR0L) ..... 1728
43.4.16. MAC address 1 high register (ENET_MAC_ADDR1H) ..... 1729
43.4.17. MAC address 1 low register (ENET_MAC_ADDR1L) ..... 1729
43.4.18. MAC address 2 high register (ENET_MAC_ADDR2H) ..... 1730
43.4.19. MAC address 2 low register (ENET_MAC_ADDR2L) ..... 1731
43.4.20. MAC address 3 high register (ENET_MAC_ADDR3H) ..... 1731
43.4.21. MAC address 3 low register (ENET_MAC_ADDR3L) ..... 1732
43.4.22. MAC flow control threshold register (ENET_MAC_FCTH) ..... 1732
43.4.23. MSC control register (ENET_MSC_CTL) ..... 1733
43.4.24. MSC receive interrupt flag register (ENET_MSC_RINTF) ..... 1734

43.4.25. MSC transmit interrupt flag register (ENET_MSC_TINTF) ..... 1735
43.4.26. MSC receive interrupt mask register (ENET_MSC_RINTMSK) ..... 1736
43.4.27. MSC transmit interrupt mask register (ENET_MSC_TINTMSK) ..... 1736
43.4.28. MSC transmitted good frames after a single collision counter register (ENET_MSC_SCCNT) 1737
43.4.29. MSC transmitted good frames after more than a single collision counter register (ENET_MSC_MSCCNT) ..... 1738
43.4.30. MSC transmitted good frames counter register (ENET_MSC_TGFCNT) ..... 1738
43.4.31. MSC received frames with CRC error counter register (ENET_MSC_RFCECNT) ..... 1739
43.4.32. MSC received frames with alignment error counter register (ENET_MSC_RFAECNT) .. 1739
43.4.33. MSC received good unicast frames counter register (ENET_MSC_RGUFCNT) ..... 1740
43.4.34. PTP time stamp control register (ENET_PTP_TSCTL) ..... 1740
43.4.35. PTP subsecond increment register (ENET_PTP_SSINC) ..... 1743
43.4.36. PTP time stamp high register (ENET_PTP_TSH) ..... 1743
43.4.37. PTP time stamp low register (ENET_PTP_TSL) ..... 1743
43.4.38. PTP time stamp update high register (ENET_PTP_TSUH) ..... 1744
43.4.39. PTP time stamp update low register (ENET_PTP_TSUL) ..... 1744
43.4.40. PTP time stamp addend register (ENET_PTP_TSADDEND) ..... 1745
43.4.41. PTP expected time high register (ENET_PTP_ETH) ..... 1745
43.4.42. PTP expected time low register (ENET_PTP_ETL) ..... 1746
43.4.43. PTP time stamp flag register (ENET_PTP_TSF) ..... 1746
43.4.44. PTP PPS control register (ENET_PTP_PPSCTL) ..... 1747
43.4.45. DMA bus control register (ENET_DMA_BCTL) ..... 1747
43.4.46. DMA transmit poll enable register (ENET_DMA_TPEN) ..... 1749
43.4.47. DMA receive poll enable register (ENET_DMA_RPEN) ..... 1750
43.4.48. DMA receive descriptor table address register (ENET_DMA_RDTADDR) ..... 1750
43.4.49. DMA transmit descriptor table address register (ENET_DMA_TDTADDR) ..... 1751
43.4.50. DMA status register (ENET_DMA_STAT) ..... 1752
43.4.51. DMA control register (ENET_DMA_CTL) ..... 1755
43.4.52. DMA interrupt enable register (ENET_DMA_INTEN) ..... 1758
43.4.53. DMA missed frame and buffer overflow counter register (ENET_DMA_MFBOCNT) ..... 1760
43.4.54. DMA receive state watchdog counter register (ENET_DMA_RSWDC) ..... 1761
43.4.55. DMA current transmit descriptor address register (ENET_DMA_CTDADDR) ..... 1761
43.4.56. DMA current receive descriptor address register (ENET_DMA_CRDADDR) ..... 1762
43.4.57. DMA current transmit buffer address register (ENET_DMA_CTBADDR) ..... 1762
43.4.58. DMA current receive buffer address register (ENET_DMA_CRBADDR) ..... 1762

**44. Comparator (CMP) ..... 1764**

**44.1. Overview ..... 1764**

**44.2. Characteristic ..... 1764**

**44.3. Function overview ..... 1764**

44.3.1. CMP clock ..... 1765
44.3.2. CMP I/O configuration ..... 1765
44.3.3. CMP operating mode ..... 1767

44.3.4. CMP Window mode ..... 1767
44.3.5. CMP hysteresis ..... 1767
44.3.6. CMP register write protection ..... 1767
44.3.7. CMP output blanking ..... 1768
44.3.8. CMP voltage scaler function ..... 1768
44.3.9. CMP interrupt ..... 1768
**44.4. Register definition ..... 1769**
44.4.1. CMP status register (CMP_STAT) ..... 1769
44.4.2. CMP interrupt flag clear register (CMP_IFC) ..... 1770
44.4.3. CMP alternate select register (CMP_SR) ..... 1770
44.4.4. CMP0 control/status register (CMP0_CS) ..... 1771
44.4.5. CMP1 control/status register (CMP1_CS) ..... 1773
**45. High-Performance Digital Filter (HPDF) ..... 1776**
**45.1. Overview ..... 1776**
**45.2. Characteristics ..... 1776**
**45.3. Function overview ..... 1777**
45.3.1. HPDF Block Diagram ..... 1777
45.3.2. HPDF on-off control ..... 1778
45.3.3. HPDF clock ..... 1778
45.3.4. Multiplex serial data channel ..... 1779
45.3.5. Parallel data input ..... 1787
45.3.6. Regular group conversion ..... 1789
45.3.7. Inserted group conversion ..... 1790
45.3.8. Digital filter ..... 1792
45.3.9. Integrator ..... 1792
45.3.10. Threshold monitor ..... 1793
45.3.11. Malfunction monitor ..... 1795
45.3.12. Extremes monitor ..... 1796
45.3.13. Data unit ..... 1796
45.3.14. HPDF interrupt ..... 1798
**45.4. Register definition ..... 1799**
45.4.1. HPDF channel x registers (x=0, 7) ..... 1799
45.4.2. HPDF filter y registers (y=0, 3) ..... 1804
**46. Real-time decryption (RTDEC) (For GD32H737_757_759) ..... 1819**
**46.1. Overview ..... 1819**
**46.2. Characteristics ..... 1819**
**46.3. Function overview ..... 1820**
46.3.1. RTDEC real-time decryption introduction ..... 1820
46.3.2. RTDEC real-time decryption introduction ..... 1821
46.3.3. Decryption with AES-128 in counter mode ..... 1821
46.3.4. RTDEC configuration ..... 1822

46.3.5. RTDEC error management ..... 1824
46.4. RTDEC interrupts ..... 1825
46.5. Registers definition ..... 1826
46.5.1. Area x configuration register (RTDEC_AREx_CFG) ..... 1826
46.5.2. Area x start address register (RTDEC_AREx_SADDR) ..... 1827
46.5.3. Area x end address register (RTDEC_AREx_EADDR) ..... 1828
46.5.4. Area x nonce register 0 (RTDEC_AREx_NONCE0) ..... 1828
46.5.5. Area x nonce register 1 (RTDEC_AREx_NONCE1) ..... 1829
46.5.6. Area x key register 0 (RTDEC_AREx_KEY0) ..... 1829
46.5.7. Area x key register 1 (RTDEC_AREx_KEY1) ..... 1830
46.5.8. Area x key register 2 (RTDEC_AREx_KEY2) ..... 1830
46.5.9. Area x key register 3 (RTDEC_AREx_KEY3) ..... 1830
46.5.10. Interrupt flag register (RTDEC_INTF) ..... 1831
46.5.11. Interrupt flag clear register (RTDEC_INTC) ..... 1832
46.5.12. Interrupt enable register (RTDEC_INTEN) ..... 1832
47. Filter arithmetic accelerator (FAC) ..... 1834
47.1. Overview ..... 1834
47.2. Characteristics ..... 1834
47.3. Function overview ..... 1834
47.3.1. General description ..... 1834
47.3.2. Local memory and buffers ..... 1835
47.3.3. Input buffers ..... 1836
47.3.4. Output buffer ..... 1838
47.3.5. Initialization functions ..... 1839
47.3.6. Filter functions ..... 1840
47.3.7. Fixed point data format ..... 1842
47.3.8. Float point data format ..... 1843
47.3.9. FIR filters ..... 1843
47.3.10. IIR filters ..... 1845
47.4. Register definition ..... 1847
47.4.1. FAC X0 buffer configure register (FAC_X0BCFG) ..... 1847
47.4.2. FAC X1 buffer configure register (FAC_X1BCFG) ..... 1847
47.4.3. FAC Y buffer configure register (FAC_YBCFG) ..... 1848
47.4.4. FAC Parameter configure register (FAC_PARACFG) ..... 1849
47.4.5. FAC Control register (FAC_CTL) ..... 1849
47.4.6. FAC Status register (FAC_STAT) ..... 1851
47.4.7. FAC write data register (FAC_WDATA) ..... 1852
47.4.8. FAC read data register (FAC_RDATA) ..... 1853
48. Hardware semaphore (HWSEM) (For GD32H737_757_759) ..... 1854
48.1. Overview ..... 1854

48.2. Characteristics ..... 1854
48.3. Function overview ..... 1854
48.3.1. Block diagram ..... 1854
48.3.2. Semaphore x ..... 1855
48.3.3. Lock a semaphore ..... 1855
48.3.4. Unlock a semaphore ..... 1856
48.3.5. Unlock all semaphores ..... 1856
48.3.6. Interrupts ..... 1857
48.4. Register definition ..... 1859
48.4.1. Control register (HWSEM_CTLx)(x = 0…31) ..... 1859
48.4.2. Read lock register (HWSEM_RLKx)(x = 0...31) ..... 1859
48.4.3. Interrupt enable register (HWSEM_INTEN) ..... 1860
48.4.4. Interrupt flag clear register (HWSEM_INTC) ..... 1861
48.4.5. Status register (HWSEM_STAT) ..... 1861
48.4.6. Interrupt flag register (HWSEM_INTF) ..... 1861
48.4.7. Unlock register (HWSEM_UNLK) ..... 1862
48.4.8. Key register (HWSEM_KEY) ..... 1862
49. Universal serial bus High-Speed interface (USBHS) ..... 1864
49.1. Overview ..... 1864
49.2. Characteristics ..... 1864
49.3. Block diagram ..... 1865
49.4. Signal description ..... 1865
49.5. Function overview ..... 1866
49.5.1. USBHS PHY selection, clocks and working modes ..... 1866
49.5.2. USB host function ..... 1869
49.5.3. USB device function ..... 1871
49.5.4. OTG function overview ..... 1873
49.5.5. Data FIFO ..... 1874
49.5.6. DMA function ..... 1877
49.5.7. Operation guide ..... 1878
49.6. Interrupts ..... 1884
49.7. Register definition ..... 1886
49.7.1. USBHS global registers ..... 1886
49.7.2. Host control and status registers ..... 1915
49.7.3. Device control and status registers ..... 1929
49.7.4. Power and clock control register (USBHS_PWRCLKCTL) ..... 1957
50. EtherCAT® SubDevice Controller (ESC) (For GD32H75E) ..... 1959
50.1. System and bus architecture ..... 1959
50.1.1. Bus architecture ..... 1959
50.1.2. Memory map ..... 1959

50.1.3. AHB direct/indirect access ..... 1962
50.1.4. Register protection in BUSY state ..... 1964
50.1.5. OPB transmission timeout function ..... 1964
50.1.6. ESC core controller (ESC_CCTL) ..... 1965
50.1.7. ESC core controller register definition ..... 1965
50.1.8. System configuration controller (ESC_SYSCFG) ..... 1971
50.1.9. System configuration register definition ..... 1971
50.2. Power management unit (PMU) ..... 1975
50.2.1. Overview ..... 1975
50.2.2. Characteristics ..... 1975
*50.2.3.* Function overview ..... 1975
50.2.4. Register definition ..... 1979
50.3. Reset and clock unit (RCU) ..... 1982
50.3.1. Reset control unit (RCTL) ..... 1982
50.3.2. Clock control unit (CCTL) ..... 1983
50.3.3. Register definition ..... 1985
50.4. Interrupt controller (INTC) ..... 1992
50.4.1. Overview ..... 1992
50.4.2. Characteristics ..... 1992
50.4.3. Interrupts function overview ..... 1992
50.4.4. Register definition ..... 1995
50.5. General-purpose I/Os (GPIO) ..... 1999
50.5.1. Overview ..... 1999
50.5.2. Characteristics ..... 1999
50.5.3. Function overview ..... 1999
50.5.4. Register definition ..... 2003
50.6. TIMER ..... 2011
50.6.1. Basic Timer ..... 2011
50.6.2. Free-Running Counter (FRC) ..... 2015
50.7. PDI Wrapper ..... 2017
50.7.1. SPI/QSPI/OSPI slave ..... 2017
50.8. Ethernet PHYS ..... 2038
50.8.1. Overview ..... 2038
50.8.2. Characteristics ..... 2038
50.8.3. Functional Overview ..... 2039
50.8.4. PHY Register definition ..... 2042
50.9. EtherCAT ..... 2086
50.9.1. Overview ..... 2086
50.9.2. Characteristics ..... 2086
50.9.3. Function overview ..... 2089
50.9.4. ESC Register definition ..... 2097

51. Appendix ..... 2162
51.1. List of abbreviations used in register ..... 2162
51.2. List of terms ..... 2162
51.3. Available peripherals ..... 2163
52. Revision history ..... 2164

# List of Figures

Figure 1-1. The structure of the Cortex®-M7 processor ..... 62
Figure 1-2. The system architecture of GD32H73x_75x devices (For GD32H737_757_759) ..... 63
Figure 1-3. The system architecture of GD32H73x_75x devices (For GD32H75E) ..... 65
Figure 1-4. Bus matrix Region 0 ..... 66
Figure 1-5. Bus matrix Region 1 ..... 67
Figure 1-6. Bus matrix Region 2 ..... 67
Figure 1-7. Block digram of AXI SRAM controller ..... 84
Figure 1-8. Block digram of RAM shared by ITCM/DTCM/AXI SRAM ..... 85
Figure 1-9. Block diagram of AXI interconnect matrix ..... 89
Figure 2-1. Block architecture of RAMECCMU ..... 140
Figure 3-1. FMC block diagram ..... 147
Figure 3-2. Proccess of sector erase operation ..... 150
Figure 3-3. Process of typical mass erase operation ..... 152
Figure 3-4. Process of protection-removed mass erase operation ..... 154
Figure 3-5. Proccess of program operation ..... 156
Figure 3-6. Memory architecture in standard mode and secure mode ..... 167
Figure 4-1. Block diagram of efuse controller ..... 191
Figure 4-2 EFUSE interrupt mapping diagram ..... 197
Figure 5-1. Power supply overview ..... 210
Figure 5-2. Waveform of the Backup domain voltage thresholds ..... 212
Figure 5-3. Waveform of the POR / PDR ..... 213
Figure 5-4. Waveform of the BOR ..... 214
Figure 5-5. Waveform of the LVD threshold ..... 214
Figure 5-6. Waveform of the VAVD threshold ..... 215
Figure 5-7. Temperature thresholds ..... 216
Figure 5-8. LDO supplies for $V_{CORE}$ power domain ..... 217
Figure 5-9. SMPS supplies for $V_{CORE}$ power domain ..... 217
Figure 5-10. Bypass ..... 218
Figure 5-11. Waveform of the VOVD ..... 219
Figure 5-12. Connection diagram of USB voltage regulator during power supply ..... 219
Figure 5-13. Connection diagram for USB voltage regulator bypass ..... 219
Figure 6-1. The system reset circuit ..... 232
Figure 6-2. Clock tree ..... 233
Figure 6-3. Clock tree ..... 237
Figure 6-4. HXTAL clock source ..... 240
Figure 6-5. HXTAL clock source in bypass mode ..... 240
Figure 7-1. CTC overview ..... 407
Figure 7-2. CTC trim counter ..... 408
Figure 8-1. Block diagram of EXTI ..... 429
Figure 9-1. TRIGSEL main composition example ..... 440

**Figure 10-1. Basic structure of a standard I/O port bit** ..... 483
**Figure 10-2. Input configuration** ..... 485
**Figure 10-3. Output configuration** ..... 486
**Figure 10-4. Analog configuration** ..... 486
**Figure 10-5. Alternate function configuration** ..... 487
**Figure 10-6. Analog configuration for ADC for GD32H73x_75x** ..... 488
**Figure 10-7. Filtering using the sampling window** ..... 489
**Figure 10-8. Input filtering clock cycle** ..... 490
**Figure 11-1. Block diagram of CRC calculation unit** ..... 509
**Figure 12-1. TRNG block diagram** ..... 515
**Figure 13-1. DATAM No swapping and Half-word swapping** ..... 526
**Figure 13-2. DATAM Byte swapping and Bit swapping** ..... 527
**Figure 13-3. CAU diagram** ..... 528
**Figure 13-4. DES / TDES ECB encryption** ..... 529
**Figure 13-5. DES / TDES ECB decryption** ..... 530
**Figure 13-6. DES / TDES CBC encryption** ..... 531
**Figure 13-7. DES / TDES CBC decryption** ..... 532
**Figure 13-8. AES ECB encryption** ..... 533
**Figure 13-9. AES ECB decryption** ..... 534
**Figure 13-10. AES CBC encryption** ..... 535
**Figure 13-11. AES CBC decryption** ..... 536
**Figure 13-12. Counter block structure** ..... 536
**Figure 13-13. AES CTR encryption/decryption** ..... 537
**Figure 14-1. DATAM No swapping and Half-word swapping** ..... 557
**Figure 14-2. DATAM Byte swapping and Bit swapping** ..... 557
**Figure 14-3. HAU block diagram** ..... 558
**Figure 15-1. TMU block diagram** ..... 571
**Figure 16-1. Block diagram of DMA** ..... 590
**Figure 16-2. Data stream for three transfer modes** ..... 591
**Figure 16-3. Handshake mechanism** ..... 592
**Figure 16-4. Data packing/unpacking when PWIDTH = '00'** ..... 596
**Figure 16-5. Data packing/unpacking when PWIDTH = '01'** ..... 597
**Figure 16-6. Data packing/unpacking when PWIDTH = '10'** ..... 597
**Figure 16-7. DMA operation of switch-buffer mode** ..... 599
**Figure 16-8. System connection of DMA0 and DMA1** ..... 605
**Figure 17-1. Block diagram of MDMA** ..... 619
**Figure 17-2. Connections of the four modes** ..... 620
**Figure 17-3. Word, halfword, byte order exchange** ..... 622
**Figure 17-4. Data padding and alignment (source greater than destination)** ..... 622
**Figure 17-5. Data padding and alignment (source less than destination)** ..... 623
**Figure 17-6. Data packing / unpacking** ..... 623
**Figure 17-7. MDMA interrupt logic** ..... 629
**Figure 18-1. Block diagram of DMAMUX** ..... 645
**Figure 18-2. Synchronization mode** ..... 647

**Figure 18-3. Event generation** ..... 648
**Figure 19-1. Block diagram of JTAG unit** ..... 665
**Figure 20-1. ADC module block diagram** ..... 677
**Figure 20-2. Single operation mode** ..... 680
**Figure 20-3. Continuous operation mode** ..... 680
**Figure 20-4. Scan operation mode, continuous disable** ..... 681
**Figure 20-5. Scan operation mode, continuous enable** ..... 682
**Figure 20-6. Discontinuous operation mode** ..... 682
**Figure 20-7. 14-bit Data storage mode** ..... 683
**Figure 20-8. 12-bit Data storage mode** ..... 683
**Figure 20-9. 6-bit data storage mode** ..... 684
**Figure 20-10. Schematic diagram of handshake signal between HPDF and ADC module** 687
**Figure 20-11. 20-bit to 16-bit result truncation (for 12bit ADC)** ..... 689
**Figure 20-12. Numerical example with 5-bits shift and rounding (for 12bit ADC)** ..... 689
**Figure 20-13. 14bit ADC oversampling with 10bits right shift** ..... 689
**Figure 20-14. Numerical example for 14bit ADC oversampling with 10bits right shift** ..... 690
**Figure 20-15. ADC sync block diagram** ..... 691
**Figure 20-16. Routine parallel mode on 16 channels** ..... 692
**Figure 20-17. Routine follow-up mode on 1 channel in continuous operation mode** ..... 693
**Figure 21-1. DAC block diagram** ..... 721
**Figure 21-2. DAC LFSR algorithm** ..... 723
**Figure 21-3. DAC triangle noise wave** ..... 724
**Figure 21-4. DAC sample and keep** ..... 727
**Figure 22-1. Free watchdog block diagram** ..... 742
**Figure 22-2. Window watchdog timer block diagram** ..... 748
**Figure 22-3. Window watchdog timing diagram** ..... 749
**Figure 23-1. Block diagram of RTC** ..... 754
**Figure 24-1. Advanced timer block diagram** ..... 790
**Figure 24-2. Normal mode, internal clock divided by 1** ..... 791
**Figure 24-3. Counter timing diagram with prescaler division change from 1 to 2** ..... 792
**Figure 24-4. Timing diagram of up counting mode, PSC=0/2** ..... 793
**Figure 24-5. Timing diagram of up counting mode, change TIMERx_CAR ongoing** ..... 794
**Figure 24-6. Timing diagram of down counting mode, PSC=0/2** ..... 795
**Figure 24-7. Timing diagram of down counting mode, change TIMERx_CAR ongoing** ... 795
**Figure 24-8. Timing diagram of center-aligned counting mode** ..... 797
**Figure 24-9. Repetition counter timing diagram of center-aligned counting mode** ..... 798
**Figure 24-10. Repetition counter timing diagram of up counting mode** ..... 798
**Figure 24-11. Repetition counter timing diagram of down counting mode** ..... 799
**Figure 24-12. Input capture logic for channel 0** ..... 800
**Figure 24-13. Input capture logic for multi mode channel 0** ..... 800
**Figure 24-14. Output compare logic (when MCHxMSEL = 2'00, x=0, 1, 2, 3)** ..... 801
**Figure 24-15. Output compare logic (when MCHxMSEL = 2'11, x=0,1,2,3)** ..... 802
**Figure 24-16. Output-compare in three modes** ..... 803
**Figure 24-17. Timing diagram of EAPWM** ..... 804

Figure 24-18. Timing diagram of CAPWM ..... 805
Figure 24-19. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD) ..... 807
Figure 24-20. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD) ..... 807
Figure 24-21. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD) ..... 807
Figure 24-22. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL ..... 808
Figure 24-23. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD ..... 808
Figure 24-24. Four Channels outputs in Composite PWM mode ..... 809
Figure 24-25. CHx_O output with a pulse in edge-aligned mode (CHxOMPSEL≠2'b00) ..... 810
Figure 24-26. CHx_O output with a pulse in center-aligned mode (CHxOMPSEL≠2'b00) 810
Figure 24-27. Complementary output with dead time insertion ..... 813
Figure 24-28. BREAK0 function logic diagram ..... 814
Figure 24-29. BREAK1 function logic diagram ..... 814
Figure 24-30. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1) ..... 815
Figure 24-31. Output behavior of the channel outputs with the BREAK0 and BREAK1 ..... 816
Figure 24-32. BRKINx (x=0...2) pins logic with BREAK0 function ..... 818
Figure 24-33. Example of counter operation in decoder interface mode ..... 819
Figure 24-34. Example of decoder interface mode with CI0FE0 polarity inverted ..... 819
Figure 24-35. Quadrature decoder signal disconnection detection block diagram ..... 820
Figure 24-36. Example of counter operation in non-quadrature decoder mode 0 with CH1P=0 ..... 820
Figure 24-37. Example of counter operation in non-quadrature decoder mode 1 with CH0P=0 ..... 821
Figure 24-38. Hall sensor is used for BLDC motor ..... 822
Figure 24-39. Hall sensor timing between two timers ..... 823
Figure 24-40. Restart mode ..... 824
Figure 24-41. Pause mode ..... 824
Figure 24-42. Event mode ..... 825
Figure 24-43. Single pulse mode TIMERx_CHxCV=0x04, TIMERx_CAR=0x60 ..... 826
Figure 24-44. delayable single pulse mode with TIMERx_CHxCV=0x00, TIMERx_CAR=0x60 ..... 827
Figure 24-45. Trigger mode of TIMER0 controlled by enable signal of TIMER2 ..... 828
Figure 24-46. Trigger mode of TIMER0 controlled by update signal of TIMER2 ..... 828
Figure 24-47. Pause mode of TIMER0 controlled by enable signal of TIMER2 ..... 829
Figure 24-48. Pause mode of TIMER0 controlled by O0CPRE signal of TIMER2 ..... 829
Figure 24-49. Trigger TIMER0 and TIMER2 by the CI0 signal of TIMER2 ..... 830
Figure 24-50. General Level 0 timer block diagram ..... 901
Figure 24-51. Normal mode, internal clock divided by 1 ..... 902
Figure 24-52. Counter timing diagram with prescaler division change from 1 to 2 ..... 903
Figure 24-53. Timing chart of up counting mode, PSC=0/2 ..... 904
Figure 24-54. Timing chart of up counting, change TIMERx_CAR ongoing ..... 904
Figure 24-55. Timing chart of down counting mode, PSC=0/2 ..... 905
Figure 24-56. Timing chart of down counting mode, change TIMERx_CAR ongoing ..... 906

Figure 24-57. Timing chart of center-aligned counting mode ..... 907
Figure 24-58. Input capture logic ..... 908
Figure 24-59. Output compare logic (x=0,1,2,3) ..... 909
Figure 24-60. Output-compare under three modes ..... 910
Figure 24-61. Timing chart of EAPWM ..... 911
Figure 24-62. Timing chart of CAPWM ..... 911
Figure 24-63. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD) ..... 913
Figure 24-64. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD) ..... 913
Figure 24-65. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD) ..... 913
Figure 24-66. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL ..... 914
Figure 24-67. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD ..... 914
Figure 24-68. Four Channels outputs in Composite PWM mode ..... 915
Figure 24-69. CHx_O output with a pulse in edge-aligned mode (CHxOMPSEL≠2'b00) ..... 916
Figure 24-70. CHx_O output with a pulse in center-aligned mode (CHxOMPSEL≠2'b00) 916
Figure 24-71. Example of counter operation in decoder interface mode ..... 918
Figure 24-72. Example of decoder interface mode with CI0FE0 polarity inverted ..... 918
Figure 24-73. Quadrature decoder signal disconnection detection block diagram ..... 919
Figure 24-74. Example of counter operation in non-quadrature decoder mode 0 with CH1P=0 ..... 919
Figure 24-75. Example of counter operation in non-quadrature decoder mode 1 with CH0P=0 ..... 920
Figure 24-76. Restart mode ..... 922
Figure 24-77. Pause mode ..... 922
Figure 24-78. Event mode ..... 922
Figure 24-79. Single pulse mode TIMERx_CHxCV = 0x04, TIMERx_CAR=0x60 ..... 924
Figure 24-80. delayable single pulse mode TIMERx_CHxCV=0x00, TIMERx_CAR=0x60 ..... 925
Figure 24-81. General level3 timer block diagram ..... 958
Figure 24-82. Normal mode, internal clock divided by 1 ..... 959
Figure 24-83. Counter timing diagram with prescaler division change from 1 to 2 ..... 960
Figure 24-84. Timing diagram of up counting mode, PSC=0/2 ..... 961
Figure 24-85. Timing diagram of up counting mode, change TIMERx_CAR on the go ..... 961
Figure 24-86. Repetition timechart for up-counter ..... 962
Figure 24-87. Input capture logic for channel 0 ..... 963
Figure 24-88. Input capture logic for multi mode channel 0 ..... 963
Figure 24-89. Output compare logic (when MCHxMSEL = 2'00, x=0) ..... 964
Figure 24-90. Output compare logic (when MCHxMSEL = 2'11, x=0) ..... 965
Figure 24-91. Output compare logic (x=1) ..... 965
Figure 24-92. Output-compare in three modes ..... 967
Figure 24-93. PWM mode timechart ..... 968
Figure 24-94. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD) ..... 970
Figure 24-95. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD) ..... 970
Figure 24-96. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD) ..... 970
Figure 24-97. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL

..... 971
Figure 24-98. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD ..... 971
Figure 24-99. CHx_O output with a pulse in edge-aligned mode (CHxOMPSEL =2'b00).. 972
Figure 24-100. Complementary output with dead-time insertion ..... 976
Figure 24-101. BREAK0 function logic diagram ..... 977
Figure 24-102. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1) ..... 977
Figure 24-103. BRKIN0 pin logic with BREAK0 function ..... 979
Figure 24-104. Restart mode ..... 980
Figure 24-105. Pause mode ..... 980
Figure 24-106. Event mode ..... 981
Figure 24-107. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60 ..... 982
Figure 24-108. delayable single pulse mode TIMERx_CHxCV=0x00, TIMERx_CAR=0x60 983
Figure 24-109. General level4 timer block diagram ..... 1018
Figure 24-110. Normal mode, internal clock divided by 1 ..... 1019
Figure 24-111. Counter timing diagram with prescaler division change from 1 to 2 ..... 1019
Figure 24-112. Timing diagram of up counting mode, PSC=0/2 ..... 1020
Figure 24-113. Timing diagram of up counting mode, change TIMERx_CAR on the go 1021
Figure 24-114. Repetition timechart for up-counter ..... 1022
Figure 24-115. Input capture logic for channel 0 ..... 1023
Figure 24-116. Input capture logic for multi mode channel 0 ..... 1023
Figure 24-117. Output compare logic (when MCHxMSEL = 2'00, x=0) ..... 1024
Figure 24-118. Output compare logic (when MCHxMSEL = 2'11, x=0) ..... 1025
Figure 24-119. Output-compare in three modes ..... 1026
Figure 24-120. PWM mode timechart ..... 1027
Figure 24-121. Complementary output with dead-time insertion ..... 1030
Figure 24-122. BREAK0 function logic diagram ..... 1031
Figure 24-123. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1) ..... 1031
Figure 24-124. BRKIN0 pin logic with BREAK0 function ..... 1033
Figure 24-125. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60 ..... 1033
Figure 24-126. Basic timer block diagram ..... 1061
Figure 24-127. Normal mode, internal clock divided by 1 ..... 1062
Figure 24-128. Counter timing diagram with prescaler division change from 1 to 2 ..... 1062
Figure 24-129. Timing chart of up counting mode, PSC=0/2 (TIMERx, x=5,6) ..... 1063
Figure 24-130. Timing chart of up counting mode, change TIMERx_CAR ongoing (TIMERx, x=5,6) ..... 1063
Figure 25-1. USART module block diagram ..... 1074
Figure 25-2. USART character frame (8 bits data and 1 stop bit) ..... 1075
Figure 25-3. USART transmit procedure ..... 1077
Figure 25-4. Oversampling method of a receive frame bit (OSB = 0) ..... 1078
Figure 25-5. Configuration step when using DMA for USART transmission ..... 1079
Figure 25-6. Configuration step when using DMA for USART reception ..... 1080
Figure 25-7. Hardware flow control between two USARTs ..... 1080

**Figure 25-8. Hardware flow control** ..... 1081
**Figure 25-9. Break frame occurs during idle state** ..... 1083
**Figure 25-10. Break frame occurs during a frame** ..... 1083
**Figure 25-11. Example of USART in synchronous mode** ..... 1084
**Figure 25-12. 8-bit format USART synchronous waveform (CLEN = 1)** ..... 1084
**Figure 25-13. IrDA SIR ENDEC module** ..... 1085
**Figure 25-14. IrDA data modulation** ..... 1085
**Figure 25-15. ISO7816-3 frame format** ..... 1086
**Figure 25-16. USART receive FIFO structure** ..... 1089
**Figure 25-17. USART transmitter FIFO structure** ..... 1089
**Figure 25-18. USART interrupt mapping diagram** ..... 1091
**Figure 26-1. I2C module block diagram** ..... 1116
**Figure 26-2. Data validation** ..... 1117
**Figure 26-3. START and STOP condition** ..... 1118
**Figure 26-4. I2C communication flow with 10-bit address (Master Transmit)** ..... 1118
**Figure 26-5. I2C communication flow with 7-bit address (Master Transmit)** ..... 1119
**Figure 26-6. I2C communication flow with 7-bit address (Master Receive)** ..... 1119
**Figure 26-7. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0)** ..... 1119
**Figure 26-8. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)** ..... 1119
**Figure 26-9. Data hold time** ..... 1120
**Figure 26-10. Data setup time** ..... 1121
**Figure 26-11. Data transmission** ..... 1123
**Figure 26-12. Data reception** ..... 1123
**Figure 26-13. I2C initialization in slave mode** ..... 1126
**Figure 26-14. Programming model for slave transmitting when SS=0** ..... 1127
**Figure 26-15. Programming model for slave transmitting when SS=1** ..... 1128
**Figure 26-16. Programming model for slave receiving** ..... 1129
**Figure 26-17. I2C initialization in master mode** ..... 1130
**Figure 26-18. Programming model for master transmitting (N<=255)** ..... 1131
**Figure 26-19. Programming model for master transmitting (N>255)** ..... 1132
**Figure 26-20. Programming model for master receiving (N<=255)** ..... 1133
**Figure 26-21. Programming model for master receiving (N>255)** ..... 1134
**Figure 26-22. SMBus master transmitter and slave receiver communication flow** ..... 1138
**Figure 26-23. SMBus master receiver and slave transmitter communication flow** ..... 1138
**Figure 27-1. Block diagram of SPI** ..... 1156
**Figure 27-2. SPI timing diagram in normal mode** ..... 1158
**Figure 27-3. SPI3 / 4 timing diagram in Quad-SPI mode (CKPL = 1, CKPH = 1, LF = 0)** ..... 1159
**Figure 27-4. SPI data frame right-aligned diagram** ..... 1159
**Figure 27-5. SPI data and clock transmission path diagram** ..... 1160
**Figure 27-6. SPI master rx delay configuration diagram** ..... 1160
**Figure 27-7. SPI slave rx delay configuration diagram** ..... 1161
**Figure 27-8. NSS signal delay timing diagram (MSSD[3:0] = 0011 (3 x Tclk), MDFD = 0011 (3**

x Tclk)) ..... 1165
Figure 27-9. NSS interlaced pulses timing diagram (MSSD[3:0] = 0011 (3 x Tclk), MDFD = 0011 (3 x Tclk)) ..... 1166
Figure 27-10. A typical full-duplex connection ..... 1167
Figure 27-11. A typical simplex connection (Master: Receive, Slave: Transmit) ..... 1167
Figure 27-12. A typical simplex connection (Master: Transmit only, Slave: Receive) ..... 1168
Figure 27-13. A typical bidirectional connection ..... 1168
Figure 27-14. Timing diagram of TI master mode with discontinuous transfer ..... 1171
Figure 27-15. Timing diagram of TI master mode with continuous transfer ..... 1171
Figure 27-16. Timing diagram of TI slave mode ..... 1171
Figure 27-17. Timing diagram of quad write operation in Quad-SPI mode ..... 1173
Figure 27-18. Timing diagram of quad read operation in Quad-SPI mode ..... 1174
Figure 27-19. Block diagram of I2S ..... 1180
Figure 27-20. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0) 1181
Figure 27-21. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1) 1181
Figure 27-22. I2S Philips standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0) 1181
Figure 27-23. I2S Philips standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1) 1182
Figure 27-24. I2S Philips standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0) 1182
Figure 27-25. I2S Philips standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1) 1182
Figure 27-26. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0) 1182
Figure 27-27. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1) 1182
Figure 27-28. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0) ..... 1182
Figure 27-29. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1) ..... 1183
Figure 27-30. MSB justified standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0) ..... 1183
Figure 27-31. MSB justified standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1) ..... 1183
Figure 27-32. MSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0) ..... 1183
Figure 27-33. MSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1) ..... 1183
Figure 27-34. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0) ..... 1183
Figure 27-35. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1) ..... 1184
Figure 27-36. LSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0) ..... 1184
Figure 27-37. LSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1) ..... 1184
Figure 27-38. LSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0) ..... 1184
Figure 27-39. LSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)

.....1184
**Figure 27-40. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0)** .....1185
**Figure 27-41. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1)** .....1185
**Figure 27-42. PCM standard short frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0)** .....1185
**Figure 27-43. PCM standard short frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1)** .....1185
**Figure 27-44. PCM standard short frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)** .....1186
**Figure 27-45. PCM standard short frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)** .....1186
**Figure 27-46. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)** .....1186
**Figure 27-47. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)** .....1186
**Figure 27-48. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0)** .....1186
**Figure 27-49. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1)** .....1187
**Figure 27-50. PCM standard long frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0)** .....1187
**Figure 27-51. PCM standard long frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1)** .....1187
**Figure 27-52. PCM standard long frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)** .....1187
**Figure 27-53. PCM standard long frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)** .....1187
**Figure 27-54. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)** .....1188
**Figure 27-55. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)** .....1188
**Figure 27-56. Block diagram of I2S clock generator** .....1188
**Figure 29-1 OSPI octal communication mode block diagram** ..... 1219
**Figure 29-2 OSPI quad communication mode block diagram** ..... 1219
**Figure 29-3 OSPI command format in octal mode** ..... 1220
**Figure 29-4 OSPI command format in quad mode** ..... 1220
**Figure 29-5 CSN and SCK behavior** ..... 1222
**Figure 30-1. Schematic diagram of CPDM** ..... 1249
**Figure 30-2. CPDM delay line length configuration flowchart** ..... 1250
**Figure 30-3. CPDM output clock phase configuration flowchart** ..... 1251
**Figure 31-1. DCI module block diagram** ..... 1254
**Figure 31-2. Hardware synchronization mode** ..... 1255

**Figure 31-3. Hardware synchronization mode: JPEG format supporting** ..... 1256
**Figure 32-1. TLI module block diagram** ..... 1271
**Figure 32-2. Display timing diagram** ..... 1271
**Figure 32-3. Block diagram of Blending** ..... 1274
**Figure 33-1. RSPDIF block diagram** ..... 1291
**Figure 33-2. S/PDIF block and S/PDIF sub-frame format** ..... 1293
**Figure 33-3. Information bits coding example** ..... 1293
**Figure 33-4. S/PDIF Preambles** ..... 1294
**Figure 33-5. Noise filtering and rising/falling edge detection** ..... 1295
**Figure 33-6. Waveforms of TWCNT** ..... 1296
**Figure 33-7. Synchronization flowchart flow** ..... 1298
**Figure 33-8. Synchronization process scheduling** ..... 1299
**Figure 33-9. RSPDIF states** ..... 1300
**Figure 33-10. RSPDIF_DATA register format** ..... 1304
**Figure 33-11. Channel/user data format** ..... 1305
**Figure 33-12. RSPDIF overrun error when RXSTEOMEN = 0 and RXDF[1:0] = 2'b0x** ..... 1309
**Figure 33-13. RSPDIF overrun error when RXSTEOMEN = 0 and RXDF[1:0] = 2'b10** ..... 1310
**Figure 33-14. RSPDIF overrun error when RXSTEOMEN = 1 and RXDF[1:0] = 2'b0x** ..... 1310
**Figure 33-15. RSPDIF overrun error when RXSTEOMEN = 1 and RXDF[1:0] = 2'b10** ..... 1312
**Figure 34-1. block diagram** ..... 1325
**Figure 34-2. Clock divider logic** ..... 1326
**Figure 34-3 FS active width** ..... 1330
**Figure 34-4 FS polarity** ..... 1330
**Figure 34-5 FS function** ..... 1330
**Figure 34-6 Slot activation** ..... 1331
**Figure 34-7 Slot distribution when FUNC = 0** ..... 1332
**Figure 34-8 Slot distribution when FUNC = 1** ..... 1332
**Figure 34-9 Slot partition convention** ..... 1332
**Figure 34-10 Offset region treatment** ..... 1333
**Figure 34-11 SD output management** ..... 1333
**Figure 34-12 Data configuration** ..... 1334
**Figure 34-13. PDM typical connection and timing** ..... 1335
**Figure 34-14. PDM data processing diagram** ..... 1336
**Figure 34-15. The start-up process of PDM data transmission** ..... 1337
**Figure 34-16. Eight microphones, under different slot widths, BxDATA register data format** ..... 1337
**Figure 34-17. Four microphones, under different slot widths, BxDATA register data format** ..... 1338
**Figure 34-18. Dual microphones, under different slot widths, BxDATA register data format** ..... 1339
**Figure 34-19 AC'97 slot partition** ..... 1341
**Figure 34-20 AC'97 tag definition** ..... 1342
**Figure 34-21 SPDIF data formate** ..... 1343
**Figure 34-22 Mute frame activation** ..... 1345

**Figure 34-23 Compander data path** ..... 1346
**Figure 34-24 Frame synchronization advanced detection** ..... 1350
**Figure 34-25 Frame synchronization postponed detection** ..... 1351
**Figure 35-1. IPA block diagram** ..... 1370
**Figure 35-2. Pixel extension from 'RGB888' to 'ARGB8888'** ..... 1375
**Figure 35-3. Pixel extension from 'RGB565' to 'ARGB8888'** ..... 1375
**Figure 35-4. Pixel extension from 'ARGB1555' or 'ARGB4444' to 'ARGB8888'** ..... 1375
**Figure 35-5. Sampling method of decimation filter** ..... 1377
**Figure 35-6. Bilinear Scaling diagram** ..... 1378
**Figure 35-7. Pixel compression** ..... 1380
**Figure 35-8. Inter timer operation** ..... 1381
**Figure 35-9. System connection of IPA** ..... 1388
**Figure 36-1. SDIO "no response" and "no data" operations** ..... 1414
**Figure 36-2. SDIO multiple blocks read operation** ..... 1414
**Figure 36-3. SDIO multiple blocks write operation** ..... 1414
**Figure 36-4. SDIO sequential read operation** ..... 1415
**Figure 36-5. SDIO sequential write operation** ..... 1415
**Figure 36-6. SDIO block diagram** ..... 1416
**Figure 36-7. CRU unit** ..... 1423
**Figure 36-8. Hardware flow timing** ..... 1427
**Figure 36-9. Command Token Format** ..... 1428
**Figure 36-10. Response Token Format** ..... 1441
**Figure 36-11. 1-bit data bus width** ..... 1444
**Figure 36-12. 4-bit data bus width** ..... 1444
**Figure 36-13. 8-bit data bus width** ..... 1445
**Figure 36-14. 4-bit data bus DDR** ..... 1445
**Figure 36-15. 8-bit data bus DDR** ..... 1446
**Figure 36-16. Boot operation timing** ..... 1456
**Figure 36-17. Alternative Boot operation timing** ..... 1457
**Figure 36-18. CMD5 timing** ..... 1466
**Figure 36-19. CMD12 stream timing** ..... 1468
**Figure 36-20. CMD11 voltage switch sequence** ..... 1469
**Figure 37-1. CFP Management Interface Architecture** ..... 1491
**Figure 37-2. MDIO Architecture** ..... 1491
**Figure 37-3. MDIO Frame Structure** ..... 1492
**Figure 37-4. MDIO slave communication process** ..... 1494
**Figure 38-1. The EXMC block diagram** ..... 1503
**Figure 38-2. EXMC memory banks** ..... 1505
**Figure 38-3. Four regions of bank0 address mapping** ..... 1506
**Figure 38-4. NAND address mapping** ..... 1507
**Figure 38-5. Diagram of bank2 common space** ..... 1508
**Figure 38-6. SDRAM address mapping** ..... 1509
**Figure 38-7. Mode 1 read access** ..... 1513
**Figure 38-8. Mode 1 write access** ..... 1514

**Figure 38-9. Mode A read access** ..... 1515
**Figure 38-10. Mode A write access** ..... 1515
**Figure 38-11. Mode 2/B read access** ..... 1517
**Figure 38-12. Mode 2 write access** ..... 1517
**Figure 38-13. Mode B write access** ..... 1517
**Figure 38-14. Mode C read access** ..... 1519
**Figure 38-15. Mode C write access** ..... 1519
**Figure 38-16. Mode D read access** ..... 1520
**Figure 38-17. Mode D write access** ..... 1521
**Figure 38-18. Multiplex mode read access** ..... 1522
**Figure 38-19. Multiplex mode write access** ..... 1522
**Figure 38-20. Read access timing diagram under async-wait signal assertion** ..... 1524
**Figure 38-21. Write access timing diagram under async-wait signal assertion** ..... 1524
**Figure 38-22. Read timing of synchronous multiplexed burst mode** ..... 1526
**Figure 38-23. Write timing of synchronous multiplexed burst mode** ..... 1527
**Figure 38-24. Access timing of common memory space of NAND flash controller** ..... 1530
**Figure 38-25. Access to none "NCE don't care" NAND Flash** ..... 1531
**Figure 38-26. SDRAM controller block diagram** ..... 1534
**Figure 38-27. Burst read operation** ..... 1538
**Figure 38-28. Data sampling clock delay chain** ..... 1538
**Figure 38-29. Burst write operation** ..... 1539
**Figure 38-30. Read access when FIFO not hit (BRSTRD=1, CL=2, SDCLK=2, PIPED=2)** 1540
**Figure 38-31. Read access when FIFO hit (BRSTRD=1)** ..... 1540
**Figure 38-32. Cross boundary read operation** ..... 1541
**Figure 38-33. Cross boundary write operation** ..... 1541
**Figure 38-34. Process for self-refresh entry and exit** ..... 1542
**Figure 38-35. Process for power-down entry and exit** ..... 1543
**Figure 39-1.Precision reference connection** ..... 1561
**Figure 40-1. LPDTS block diagram** ..... 1565
**Figure 40-2. Method for different REF_CLK** ..... 1567
**Figure 41-1. Block diagram of EDOUT** ..... 1578
**Figure 41-2. ABZ-phase output waveforms** ..... 1579
**Figure 41-3. Example of the settings of EDOUT and the AB-Phase and Z-Phase output waveforms** ..... 1581
**Figure 42-1. CAN module block diagram** ..... 1588
**Figure 42-2. Transmitter delay** ..... 1619
**Figure 42-3. CAN bit time** ..... 1622
**Figure 43-1. ENET module block diagram** ..... 1658
**Figure 43-2. MAC / Tagged MAC frame format** ..... 1660
**Figure 43-3. Station management interface signals** ..... 1662
**Figure 43-4. Media independent interface signals** ..... 1664
**Figure 43-5. Reduced media-independent interface signals** ..... 1666
**Figure 43-6. Descriptor ring and chain structure** ..... 1678
**Figure 43-7. Transmit descriptor in normal mode** ..... 1683

**Figure 43-8. Transmit descriptor in enhanced mode** ..... 1689
**Figure 43-9. Receive descriptor in normal mode** ..... 1693
**Figure 43-10. Receive descriptor in enhanced mode** ..... 1699
**Figure 43-11. Wakeup frame filter register** ..... 1704
**Figure 43-12. System time update using the fine correction method** ..... 1706
**Figure 43-13. MAC interrupt scheme** ..... 1712
**Figure 43-14. Ethernet interrupt scheme** ..... 1712
**Figure 43-15. Wakeup frame filter register** ..... 1723
**Figure 44-1. CMP block diagram** ..... 1765
**Figure 44-2. CMP hysteresis** ..... 1767
**Figure 44-3. The CMP outputs signal blanking** ..... 1768
**Figure 45-1. HPDF block diagram** ..... 1777
**Figure 45-2. The sequence diagram of SPI data transmission** ..... 1781
**Figure 45-3. The sequence diagram of Manchester data transmission** ..... 1782
**Figure 45-4. Manchester synchronous sequence diagram** ..... 1782
**Figure 45-5. Clock loss detection timing diagram** ..... 1785
**Figure 45-6. Channel pins redirection** ..... 1786
**Figure 45-7. HPDF module external input data processing flow** ..... 1796
**Figure 45-8. HPDF interrupt logic diagram** ..... 1798
**Figure 46-1. RTDEC block diagram (for RTDEC0 and RTDEC1)** ..... 1820
**Figure 46-2. Decryption flow with AES-128 in CTR mode** ..... 1822
**Figure 46-3. RTDEC interrupt mapping diagram** ..... 1825
**Figure 47-1 FAC structure diagram** ..... 1835
**Figure 47-2 Input buffer area** ..... 1836
**Figure 47-3 Circular input buffer area** ..... 1837
**Figure 47-4 Circular input buffer operation** ..... 1838
**Figure 47-5 Circular output buffer** ..... 1838
**Figure 47-6 Circular output buffer area** ..... 1839
**Figure 47-7 The structure of FIR filter function** ..... 1841
**Figure 47-8 The structure of IIR filter** ..... 1842
**Figure 48-1. HWSEM diagram** ..... 1854
**Figure 48-2. HWSEM state machine** ..... 1855
**Figure 48-3. First try lock success** ..... 1857
**Figure 48-4. Second try lock success** ..... 1858
**Figure 48-5. Second try lock fail** ..... 1858
**Figure 48-6. HWSEM interrupt logic** ..... 1858
**Figure 49-1. USBHS block diagram** ..... 1865
**Figure 49-2. Connection using internal embedded PHY with host or device mode** ..... 1867
**Figure 49-3. Connection using internal embedded PHY with OTG mode** ..... 1868
**Figure 49-4. Connection using external ULPI PHY** ..... 1868
**Figure 49-5. State transition diagram of host port** ..... 1869
**Figure 49-6. Host mode FIFO space in SRAM** ..... 1875
**Figure 49-7. Host mode FIFO access register map** ..... 1875
**Figure 49-8. Device mode FIFO space in SRAM** ..... 1876

**Figure 49-9. Device mode FIFO access register map** ..... 1877
**Figure 50-1. Bus architecture of ESC** ..... 1959
**Figure 50-2. PME interrupt pending** ..... 1976
**Figure 50-3. Clock tree** ..... 1984
**Figure 50-4. HXTAL clock source** ..... 1985
**Figure 50-5. Block diagram of interrupt** ..... 1993
**Figure 50-6. Port line PHYS** ..... 2001
**Figure 50-7. Basic structure of Analog configuration** ..... 2002
**Figure 50-8. Basic structure of Alternate function configuration** ..... 2003
**Figure 50-9. Basic timer block diagram** ..... 2012
**Figure 50-10. Timing chart of down counting mode** ..... 2013
**Figure 50-11. FRC block diagram** ..... 2015
**Figure 50-12. Block diagram of PDI wrapper** ..... 2017
**Figure 50-13. Block diagram of SPI** ..... 2018
**Figure 50-14. Enable QSPI** ..... 2023
**Figure 50-15. Enable OSPI** ..... 2023
**Figure 50-16. SPI MODE RESET SPI** ..... 2024
**Figure 50-17. QSPI MODE RESET QSPI** ..... 2024
**Figure 50-18. OSPI MODE RESET OSPI** ..... 2024
**Figure 50-19. SPI READ** ..... 2026
**Figure 50-20. QSPI READ** ..... 2026
**Figure 50-21. OSPI READ** ..... 2027
**Figure 50-22. SPI DUAL OUTPUT READ** ..... 2028
**Figure 50-23. SPI QUAD OUTPUT READ** ..... 2029
**Figure 50-24. SPI DUAL I/O READ** ..... 2030
**Figure 50-25. SPI QUAD I/O READ** ..... 2031
**Figure 50-26. SPI WRITE** ..... 2032
**Figure 50-27. QSPI WRITE** ..... 2033
**Figure 50-28. OSPI WRITE** ..... 2033
**Figure 50-29. SPI DUAL DATA WRITE** ..... 2034
**Figure 50-30. SPI QUAD DATA WRITE** ..... 2035
**Figure 50-31. SPI DUAL ADDRESS / DATA WRITE** ..... 2036
**Figure 50-32. SPI QUAD ADDRESS / DATA WRITE** ..... 2037
**Figure 50-33.PHY functional block diagram** ..... 2039
**Figure 50-34. LED connect diagram** ..... 2041
**Figure 50-35. 100Base-TX LPI** ..... 2042
**Figure 50-36. EtherCAT system block diagram** ..... 2087
**Figure 50-37. EtherCAT State Machine** ..... 2094
**Figure 50-38. EEPROM Layout** ..... 2095
**Figure 50-39. PDI Interrupt Masking and interrupt signals** ..... 2096
**Figure 50-40. EtherCAT Interrupt Masking** ..... 2097

# List of Tables

Table 1-1. The interconnection relationship of the interconnect matrix ..... 63
Table 1-2. Memory map of GD32H73x_75x devices (For GD32H737_757_759) ..... 68
Table 1-3. Memory map of GD32H73x_75x devices (For GD32H75E) ..... 75
Table 1-4. Configuration of ITCM/DTCM/AXI SRAM ..... 85
Table 1-5. Boot mode selection ..... 86
Table 1-6. Details of Boot mode (For GD32H737_757_759) ..... 86
Table 1-7. Details of Boot mode (For GD32H75E) ..... 87
Table 1-8. Configuration of ASIBs ..... 89
Table 1-9. Configuration of AMIBs ..... 90
Table 2-1. RAMECC monitor x unit for Region 0 (x=0..4) ..... 140
Table 2-2. RAMECC monitor x unit for Region 1 (x=0..2) ..... 141
Table 3-1. GD32H73x_75x base address and size for flash memory ..... 148
Table 3-2. Option byte ..... 158
Table 3-3. WP bit for sectors protected ..... 161
Table 3-4. SPC protection level configuration ..... 163
Table 3-5. DCRP area configuration ..... 164
Table 3-6. Secure user area configuration ..... 166
Table 3-7. Function resetAndInitializeSecureAreas ..... 168
Table 3-8. Function exitSecureArea ..... 169
Table 3-9 FMC interrupt requests ..... 171
Table 4-1. System parameters ..... 194
Table 4-2. EFUSE interrupt requests ..... 197
Table 5-1. Supply mode ..... 218
Table 5-2. Power saving mode summary ..... 221
Table 6-1. Clock output 0 source select ..... 244
Table 6-2. Clock output 1 source select ..... 245
Table 8-1. NVIC exception types in Cortex®-M7 ..... 418
Table 8-2. Interrupt vector table (For GD32H737_757_759) ..... 418
Table 8-3. Interrupt vector table (For GD32H75E) ..... 423
Table 8-4. EXTI source (For GD32H737_757_759) ..... 430
Table 8-5. EXTI source (For GD32H75E) ..... 431
Table 9-1. Trigger input bit fields selection ..... 440
Table 9-2. TRIGSEL input and output mapping ..... 444
Table 10-1. GPIO configuration table ..... 483
Table 12-1. ALGO configurations ..... 516
Table 15-1. Input data configuration ..... 572
Table 15-2. Output data configuration ..... 573
Table 15-3. TMU mode configuration ..... 573
Table 15-4. Mode 0 description ..... 573
Table 15-5. Mode 1 description ..... 575
Table 15-6. Mode 2 description ..... 576

**Table 15-7. Mode 3 description** ..... 577
**Table 15-8. Mode 4 description** ..... 578
**Table 15-9. Mode 5 description** ..... 578
**Table 15-10. Mode 6 description** ..... 579
**Table 15-11. Mode 7 description** ..... 580
**Table 15-12. Mode 8 description** ..... 581
**Table 15-13. Recommended scaling factors in mode 8** ..... 581
**Table 15-14. Mode 9 description** ..... 581
**Table 15-15. Recommended scaling factors in mode 9** ..... 582
**Table 15-16. Precision vs. number of iterations** ..... 582
**Table 16-1. Transfer mode** ..... 591
**Table 16-2. CNT configuration** ..... 594
**Table 16-3. FIFO counter critical value configuration rules** ..... 595
**Table 16-4. DMA interrupt events** ..... 602
**Table 17-1. Transfer mode** ..... 619
**Table 17-2. MDMA hardware request sources** ..... 620
**Table 17-3. Source and destination address generation configuration** ..... 624
**Table 17-4. Update mode of source and destination address** ..... 627
**Table 17-5. Register link address** ..... 627
**Table 17-6. MDMA error flags** ..... 628
**Table 17-7. MDMA interrupt events** ..... 629
**Table 18-1. Interrupt events** ..... 650
**Table 18-2. Request multiplexer input mapping** ..... 651
**Table 18-3. Trigger input mapping** ..... 656
**Table 18-4. Synchronization input mapping** ..... 657
**Table 19-1. Pin assignment** ..... 664
**Table 20-1. ADC internal input signals** ..... 676
**Table 20-2. ADC input pins definition** ..... 676
**Table 20-3. ADC differential channel pin matching** ..... 678
**Table 20-4. Trigger source for routine channels for ADC0/ADC1/ADC2** ..... 684
**Table 20-5. $t_{CONV}$ timings depending on resolution for ADC0 and ADC1** ..... 687
**Table 20-6. $t_{CONV}$ timings depending on resolution for ADC2** ..... 687
**Table 20-7. Some examples show the maximum output results for N and M combimations (grayed values indicates truncation，for 12bit ADC)** ..... 690
**Table 20-8. ADC sync mode table** ..... 691
**Table 21-1. DAC I/O description** ..... 721
**Table 21-2. DAC triggers and outputs summary** ..... 721
**Table 21-3. Triggers of DAC** ..... 722
**Table 21-4. Formula of sample and refresh time** ..... 726
**Table 22-1. Min/max FWDGT timeout period at 32KHz (IRC32K)** ..... 743
**Table 22-2. Min-max timeout value at 150 MHz ($f_{PCLK3}$)** ..... 749
**Table 23-1 RTC pin configuration and function** ..... 765
**Table 23-2 RTC power saving mode management** ..... 765
**Table 23-3 RTC interrupts control** ..... 766

**Table 24-1. Timers (TIMERx) are divided into five sorts** ..... 786
**Table 24-2. Advanced timer channel description** ..... 790
**Table 24-3.The Composite PWM pulse width** ..... 806
**Table 24-4. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)** ..... 812
**Table 24-5. Output behavior of the channel in response to a BREAK0 and BREAK1 (the break input is high active)** ..... 815
**Table 24-6. Break function input pins locked/ released conditions** ..... 817
**Table 24-7. Counting direction in different quadrature decoder signals** ..... 819
**Table 24-8. the counter operation in in non-quadrature decoder mode 1** ..... 821
**Table 24-9. Examples of slave mode** ..... 823
**Table 24-10.The Composite PWM pulse width** ..... 912
**Table 24-11. Counting direction in different quadrature decoder signals** ..... 918
**Table 24-12. the counter operation in in non-quadrature decoder mode 1** ..... 920
**Table 24-13. Examples of slave mode** ..... 921
**Table 24-14.The Composite PWM pulse width** ..... 969
**Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)** ..... 975
**Table 24-16. Break function input pins locked/ released conditions** ..... 979
**Table 24-17. Slave mode example table** ..... 979
**Table 24-18. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)** ..... 1028
**Table 24-19. Break function input pins locked/ released conditions** ..... 1032
**Table 25-1. Description of USART important pins** ..... 1074
**Table 25-2. Configuration of stop bits** ..... 1075
**Table 25-3. USART interrupt requests** ..... 1090
**Table 26-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)** ..... 1116
**Table 26-2. Data setup time and data hold time** ..... 1122
**Table 26-3. Communication modes to be shut down** ..... 1123
**Table 26-4. I2C error flags** ..... 1139
**Table 26-5. I2C interrupt events** ..... 1139
**Table 27-1. SPI signal description** ..... 1157
**Table 27-2. Quad-SPI signal description** ..... 1157
**Table 27-3. MISO / MISO signal switching description** ..... 1158
**Table 27-4. The maxinum number of data frame stored in SPIX FIFO** ..... 1161
**Table 27-5. NSS function in slave mode** ..... 1164
**Table 27-6. NSS function in master mode** ..... 1165
**Table 27-7. SPI operation modes** ..... 1166
**Table 27-8. SPI interrupt requests** ..... 1179
**Table 27-9. I2S bitrate calculation formulas** ..... 1188
**Table 27-10. Audio sampling frequency calculation formulas** ..... 1189
**Table 27-11. The maxinum number of data frame stored in I2SX FIFO** ..... 1189
**Table 27-12. Direction of I2S interface signals for each operation mode** ..... 1190
**Table 27-13. I2S interrupt** ..... 1194
**Table 28-1 OSPIM matrix mapping** ..... 1215
**Table 29-1 OSPI signal description** ..... 1217

**Table 29-2 OSPI interrupt requests** ..... 1227
**Table 31-1. PINs used by DCI** ..... 1255
**Table 31-2. Typical one-line data composition** ..... 1257
**Table 31-3. Coding for SAV and EAV** ..... 1257
**Table 31-4. A frame image format definition** ..... 1258
**Table 31-5. General case of progressive mode** ..... 1258
**Table 31-6. Memory view in byte padding mode** ..... 1260
**Table 31-7. Memory view in half-word padding mode** ..... 1260
**Table 31-8. Status/Error flags** ..... 1260
**Table 32-1. Pins of display interface provided by TLI** ..... 1271
**Table 32-2. Supported pixel formats** ..... 1273
**Table 32-3. Status flags** ..... 1275
**Table 32-4. Error flags** ..... 1275
**Table 33-1. Sub-frame pattern** ..... 1292
**Table 33-2. Preamble pattern** ..... 1294
**Table 33-3. RSPDIF time interval** ..... 1294
**Table 33-4. Transition encoder encoding rules** ..... 1297
**Table 33-5. Thresholds calculation** ..... 1297
**Table 33-6. Preamble transition sequence** ..... 1297
**Table 33-7. Bi-phase decoding rules** ..... 1297
**Table 33-8. RSPDIF state** ..... 1300
**Table 33-9. RSPDIF_CTL register access permissions in different states** ..... 1302
**Table 33-10. Mixing data and control flow** ..... 1305
**Table 33-11. Minimum rspdif_ck frequency and audio sampling rate** ..... 1306
**Table 33-12. Conditions of RSPDIF_symbol_ck generation** ..... 1307
**Table 33-13. RSPDIF interrupt events** ..... 1312
**Table 34-1. Commonly used audio sampling rate** ..... 1327
**Table 34-2. External synchronization configuration** ..... 1329
**Table 34-3 FIFO request generation conditions** ..... 1334
**Table 34-4 Get the microphone data read register times under different configurations** ..... 1337
**Table 34-5. TDM configuration** ..... 1340
**Table 34-6 AC'97 transmitter tag definition** ..... 1342
**Table 34-7 AC'97 receiver tag definition** ..... 1342
**Table 34-8. SOPD mode** ..... 1343
**Table 34-9 Parity bit calculation** ..... 1344
**Table 34-10 Mute frame composition** ..... 1345
**Table 34-11 A-law encoding** ..... 1346
**Table 34-12 A-law decoding** ..... 1347
**Table 34-13 Mu-law encoding** ..... 1347
**Table 34-14 Mu-law decoding** ..... 1348
**Table 34-15 Interrupt control** ..... 1352
**Table 35-1. Signals description of IPA** ..... 1371
**Table 35-2. IPA conversion mode** ..... 1372

**Table 35-3. Foreground and background CLUT pixel format** ..... 1373
**Table 35-4. Foreground and background pixel format** ..... 1374
**Table 35-5. Expected coefficients for YUV and YCbCr modes** ..... 1376
**Table 35-6. Alpha channel value modulation** ..... 1376
**Table 35-7. Destination pixel format** ..... 1379
**Table 35-8. IPA interrupt events** ..... 1386
**Table 36-1. 2 SDIOs** ..... 1412
**Table 36-2. SDIO operation modes for SD & SD I/O card** ..... 1415
**Table 36-3. SDIO operation modes for *e*•MMC card** ..... 1415
**Table 36-4. SDIO internal input/output singnals** ..... 1416
**Table 36-5. SDIO pins functions** ..... 1417
**Table 36-6. SDIO CMD and data phase selection** ..... 1418
**Table 36-7. SDIO connections to MDMA** ..... 1426
**Table 36-8. AHB and *e*•MMC clock frequence relation** ..... 1426
**Table 36-9. AHB and SD/SD I/O card clock frequence relation** ..... 1426
**Table 36-10. Command format** ..... 1429
**Table 36-11. Card command classes (CCCs)** ..... 1429
**Table 36-12. Basic commands (class 0)** ..... 1431
**Table 36-13. Block-Oriented read commands (class 2)** ..... 1433
**Table 36-14. Stream read commands (class 1) and stream write commands (class 3)** ..... 1434
**Table 36-15. Block-Oriented write commands (class 4)** ..... 1435
**Table 36-16. Erase commands (class 5)** ..... 1436
**Table 36-17. Block oriented write protection commands (class 6)** ..... 1437
**Table 36-18. Lock card (class 7)** ..... 1437
**Table 36-19. Application-specific commands (class 8)** ..... 1437
**Table 36-20. I/O mode commands (class 9)** ..... 1438
**Table 36-21. Switch function commands (class 10)** ..... 1439
**Table 36-22. Response R1** ..... 1441
**Table 36-23. Response R2** ..... 1442
**Table 36-24. Response R3** ..... 1442
**Table 36-25. Response R4 for e•MMC** ..... 1442
**Table 36-26. Response R4 for SD I/O** ..... 1443
**Table 36-27. Response R5 for *e*•MMC** ..... 1443
**Table 36-28. Response R5 for SD I/O** ..... 1443
**Table 36-29. Response R6** ..... 1443
**Table 36-30. Response R7** ..... 1444
**Table 36-31. Card status** ..... 1447
**Table 36-32. SD status** ..... 1450
**Table 36-33. Performance move field** ..... 1451
**Table 36-34. AU_SIZE field** ..... 1452
**Table 36-35. Maximum AU size** ..... 1452
**Table 36-36. Erase size field** ..... 1452
**Table 36-37. Erase timeout field** ..... 1453
**Table 36-38. Erase offset field** ..... 1453

**Table 36-39. Lock card data structure** ..... 1463
**Table 36-40. CMD12 instructions** ..... 1467
**Table 36-41. SDIO_RESPx register at different response type** ..... 1477
**Table 37-1. MDIO pins definition** ..... 1490
**Table 37-2. Frame Details for Different Frame Types (1)** ..... 1492
**Table 37-3. Operation Code** ..... 1493
**Table 38-1. EXMC bank mapping** ..... 1505
**Table 38-2. SDRAM mapping** ..... 1509
**Table 38-3. NOR flash interface signals description** ..... 1509
**Table 38-4. PSRAM non-muxed signal description** ..... 1510
**Table 38-5. EXMC bank 0 supports all transactions** ..... 1510
**Table 38-6. NOR / PSRAM controller timing parameters** ..... 1512
**Table 38-7. EXMC timing models** ..... 1512
**Table 38-8. Mode 1 related registers configuration** ..... 1514
**Table 38-9. Mode A related registers configuration** ..... 1515
**Table 38-10. Mode 2/B related registers configuration** ..... 1517
**Table 38-11. Mode C related registers configuration** ..... 1519
**Table 38-12. Mode D related registers configuration** ..... 1521
**Table 38-13. Multiplex mode related registers configuration** ..... 1523
**Table 38-14. Timing configurations of synchronous multiplexed read mode** ..... 1526
**Table 38-15. Timing configurations of synchronous multiplexed write mode** ..... 1527
**Table 38-16. 8-bit or 16-bit NAND interface signal** ..... 1528
**Table 38-17. Bank2 of EXMC support the memory and access mode** ..... 1529
**Table 38-18. NAND flash programmable parameters** ..... 1530
**Table 38-19. SDRAM command truth table** ..... 1534
**Table 38-20. IO definition of SDRAM controller** ..... 1535
**Table 39-1. VREF modes** ..... 1561
**Table 40-1. LPDTS signals** ..... 1566
**Table 40-2. Sampling time configuration** ..... 1567
**Table 40-3. Trigger configuration** ..... 1568
**Table 40-4. Temperature sensor behavior in low-power modes** ..... 1569
**Table 40-5. Temperature sensor behavior in low-power modes** ..... 1570
**Table 42-1. Mailbox descriptor with 64 byte payload** ..... 1589
**Table 42-2. Data bytes for DLC** ..... 1591
**Table 42-3. Mailbox Rx CODE** ..... 1591
**Table 42-4. Mailbox Tx CODE** ..... 1592
**Table 42-5. Mailbox size** ..... 1594
**Table 42-6. Rx FIFO descriptor** ..... 1595
**Table 42-7. Mailbox arbitration value(32 bit) when local priority disabled** ..... 1604
**Table 42-8. Mailbox arbitration value(35 bit) when local priority enabled** ..... 1604
**Table 42-9. Rx mailbox matching** ..... 1611
**Table 42-10. Rx FIFO matching** ..... 1612
**Table 42-11. Interrupt events** ..... 1625
**Table 42-12. Rx FIFO filter element number** ..... 1639

**Table 43-1. Ethernet signals (MII)** ..... 1660
**Table 43-2. Ethernet signals (RMII)** ..... 1661
**Table 43-3. Clock range** ..... 1663
**Table 43-4. Rx interface signal encoding** ..... 1665
**Table 43-5. Destination address filtering table** ..... 1671
**Table 43-6. Source address filtering table** ..... 1671
**Table 43-7. Error status decoding in Receive Descriptor0, only used for normal descriptor (DFM=0)** ..... 1696
**Table 43-8. Supported time stamp snapshot with PTP register configuration** ..... 1742
**Table 44-1. CMP inputs and outputs summary** ..... 1766
**Table 45-1. HPDF pins definition** ..... 1777
**Table 45-2. HPDF internal signal** ..... 1778
**Table 45-3. SPI interface clock configuration** ..... 1780
**Table 45-4. Parallel data packed mode** ..... 1789
**Table 45-5. Trigger signal of inserted group** ..... 1791
**Table 45-6. The relationship between the maximum output resolution and oversampling filtering of SincX filtering** ..... 1792
**Table 45-7. Relationship between the maximum output resolution and IOR, SFOR, SFO of the integrator** ..... 1793
**Table 45-8. Features of threshold monitor working mode** ..... 1793
**Table 45-9. Maximum output rate** ..... 1797
**Table 45-10. HPDF interrupt event** ..... 1798
**Table 46-1. RTDEC interrupt requests** ..... 1825
**Table 47-1 IEEE 32-Bit Single Precision Floating-Point Format** ..... 1843
**Table 48-1. interrupt events** ..... 1858
**Table 49-1. USBHS signal description** ..... 1865
**Table 49-2. USBHS supported speeds** ..... 1866
**Table 49-3. USBHS global interrupt** ..... 1885
**Table 50-1. Memory map** ..... 1960
**Table 50-2. Alignment of valid data** ..... 1963
**Table 50-3. Power saving mode summary** ..... 1977
**Table 50-4. GPIO configuration table** ..... 1999
**Table 50-5. 4-wire mode** ..... 2018
**Table 50-6. 6-wire mode** ..... 2018
**Table 50-7. OSPI 8-line mode** ..... 2019
**Table 50-8. SPI instructions** ..... 2021
**Table 50-9. QSPI instruction** ..... 2022
**Table 50-10. OSPI instruction** ..... 2022
**Table 50-11. PDIs for EtherCAT** ..... 2089
**Table 50-12. Registers affected by PDI register function acknowledge by write** ..... 2090
**Table 51-1. List of abbreviations used in register** ..... 2162
**Table 51-2. List of terms** ..... 2162
**Table 52-1. Revision history** ..... 2164

# 1. System and memory architecture

The devices of GD32H73x_75x series are 32-bit general-purpose microcontrollers based on the Arm® Cortex®-M7 processor. The Arm® Cortex®-M7 processor includes 64-bit AMBA4 AXI, 32-bit AHB peripheral (AHBP) port, 32-bit AHB slave port for external master to access memories and APB interface for CoreSight debug components. The memory organization uses a Harvard architecture, pre-defined memory map and up to 4 GB of memory space, making the system flexible and extendable.

## 1.1. Arm® Cortex®-M7 processor

The Arm® Cortex®-M7 processor is a highly efficient high-performance, embedded processor that features low interrupt latency, low-cost debug, and has backwards compatibility with existing Cortex-M profile processors. The processor has an in-order super-scalar pipeline that means many instructions can be dual-issued, including load/load and load/store instruction pairs because of multiple memory interfaces. The Cortex®-M7 is a high-performance processor, which features a 6-stage superscalar pipeline with branch prediction and an optional FPU capable of single-precision and optionally double-precision operations. The instruction and data buses have been enlarged to 64-bit wide over the previous 32-bit buses.

The interfaces that the processor supports include:

- 64-bit AXI4 interface
- 32-bit AHB master interface
- 32-bit AHB slave interface
- 64-bit instruction TCM interface
- 2x32-bit data TCM interfaces

The processor contains the following external interfaces:

- AHBP interface
- AHBS interface
- AHBD interface
- External Private Peripheral Bus
- ATB interfaces
- TCM interface
- Cross Trigger interface
- MBIST interface
- AXIM interface

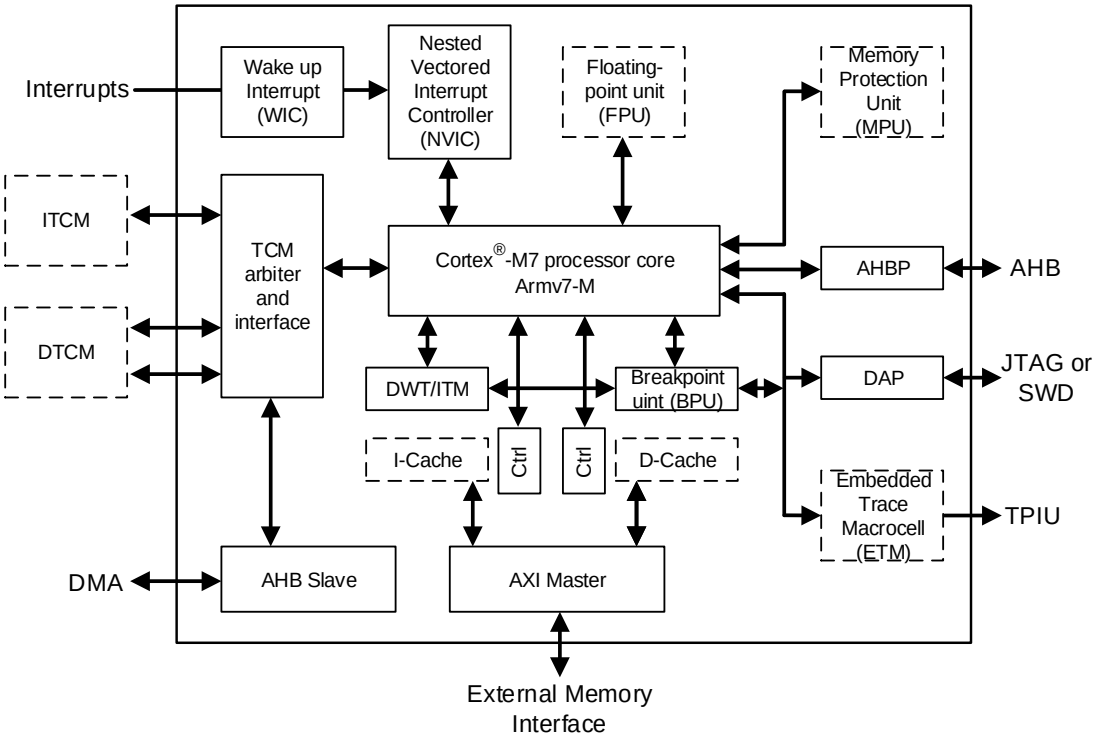
The Cortex®-M7 processor is based on the ARMv7-M architecture and supports a powerful and scalable instruction set including general data processing I/O control tasks, advanced data processing bit field manipulations, DSP and floating point instructions. Some system peripherals listed below are also provided by Cortex®-M7:

- Nested Vectored Interrupt Controller (NVIC)

- Flash Patch and Breakpoint (FPB)
- Data Watchpoint and Trace (DWT)
- Instrumentation Trace Macrocell (ITM)
- Embedded Trace Macrocell (ETM)
- JTAG or SWD Debug Port
- Trace Port Interface Unit (TPIU)
- Memory Protection Unit (MPU)
- Floating Point Unit (FPU), double-precision
- Load Store Unit (LSU)
- Data Processing Unit (DPU)
- Prefetch Unit (PFU)

***Figure 1-1. The structure of the Cortex®-M7 processor*** shows the Cortex®-M7 processor block diagram. For more information, refer to the Arm® Cortex®-M7 Technical Reference Manual.

**Figure 1-1. The structure of the Cortex®-M7 processor**




## 1.2. System architecture

An interconnect matrix includes an AXI bus matrix and two AHB bus matrices, which enables parallel access paths between multiple masters and slaves in the system. The interconnection relationship of the interconnect matrix is shown in ***Table 1-1. The interconnection relationship of the interconnect matrix***. In the following table, “1” indicates the corresponding master is able to access the corresponding slave through the interconnect matrix, while the blank means the corresponding master cannot access the corresponding

slave through the interconnect matrix.

**Table 1-1. The interconnection relationship of the interconnect matrix**

| Slave interface \ Master interface | AXIM | AHBP | ITCM | DTCM | MDMA AXI | MDMA AHBS | DMA0 MEM | DMA0 PERIPH | DMA1 MEM | DMA1 PERIPH | IPA | TLI | SDIO0 | SDIO1 | ENET | USBHS0 | USBHS1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ITCM | | | 1 | | | 1 | | | | | | | | | | | |
| DTCM | | | | 1 | | 1 | | | | | | | | | | | |
| FMC | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| AXI SRAM | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| RAM shared (ITCM/DTCM/AXI) | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SRAM0 | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | 1 |
| SRAM1 | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | | | 1 | 1 | 1 | 1 |
| Backup RAM | 1 | | | | 1 | | 1 | 1 | 1 | 1 | | | | 1 | | | 1 |
| AHB1 | 1 | 1 | | | 1 | | 1 | 1 | 1 | 1 | 1 | | | 1 | | | |
| AHB2 | | 1 | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | |
| AHB3 | 1 | | | | 1 | | | | | | | | | | | | |
| AHB4 | 1 | | | | 1 | | 1 | 1 | 1 | 1 | | | | 1 | | | 1 |
| APB1 | | 1 | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | |
| APB2 | | 1 | | | | | 1 | 1 | 1 | 1 | | | | 1 | | | |
| APB3 | 1 | | | | 1 | | | | | | | | | | | | |
| APB4 | 1 | | | | 1 | | 1 | 1 | 1 | 1 | | | | 1 | | | 1 |
| EXMC | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| OSPI | 1 | | | | 1 | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

The system architecture of GD32H73x_75x devices is shown in ***Figure 1-2. The system architecture of GD32H73x_75x devices (For GD32H737_757_759)*** and ***Figure 1-3. The system architecture of GD32H73x_75x devices (For GD32H75E)***, and the work frequency is related to the voltage of power supply, please refer to the datasheet.

**Figure 1-2. The system architecture of GD32H73x_75x devices (For**

**GD32H737_757_759)**

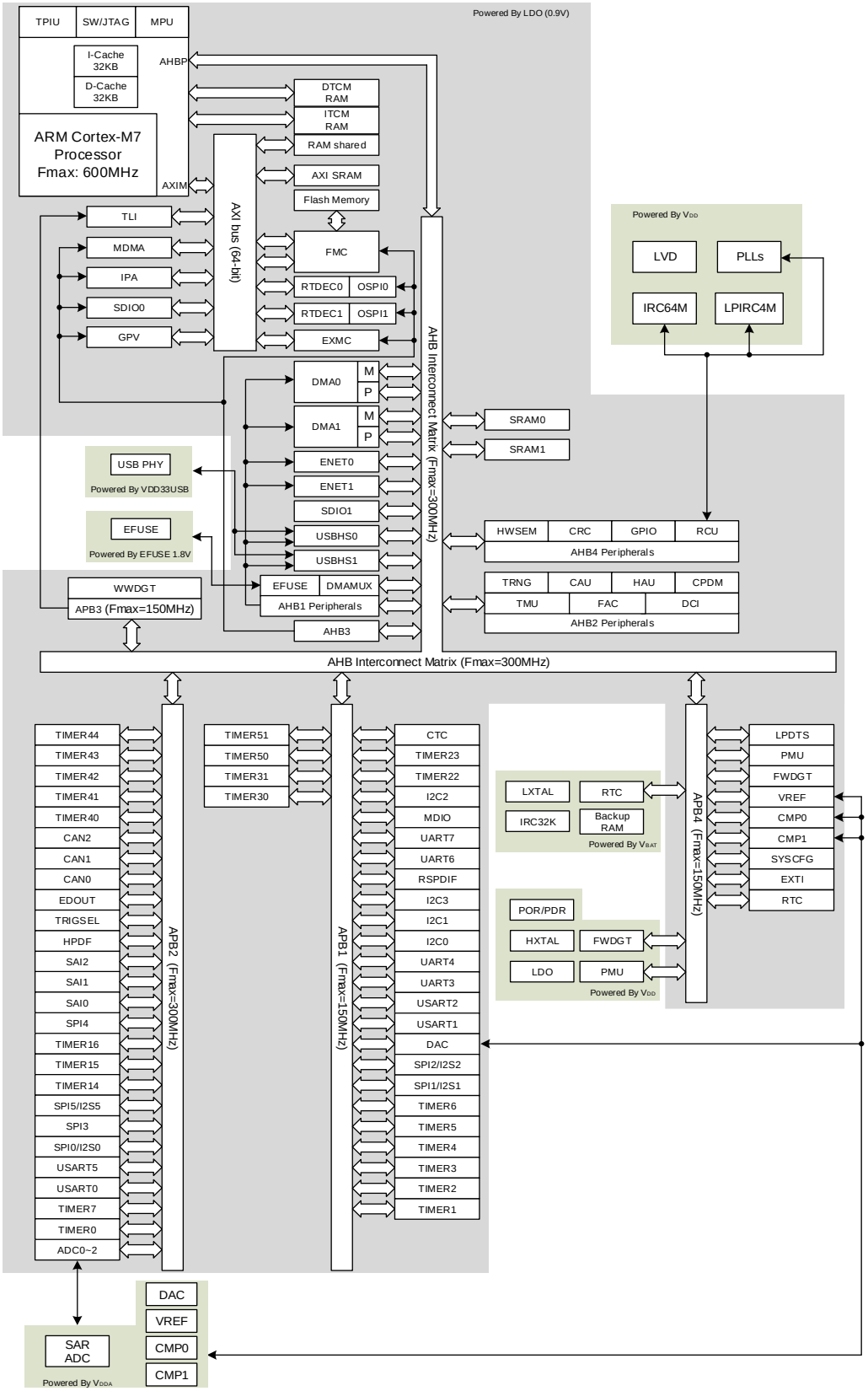
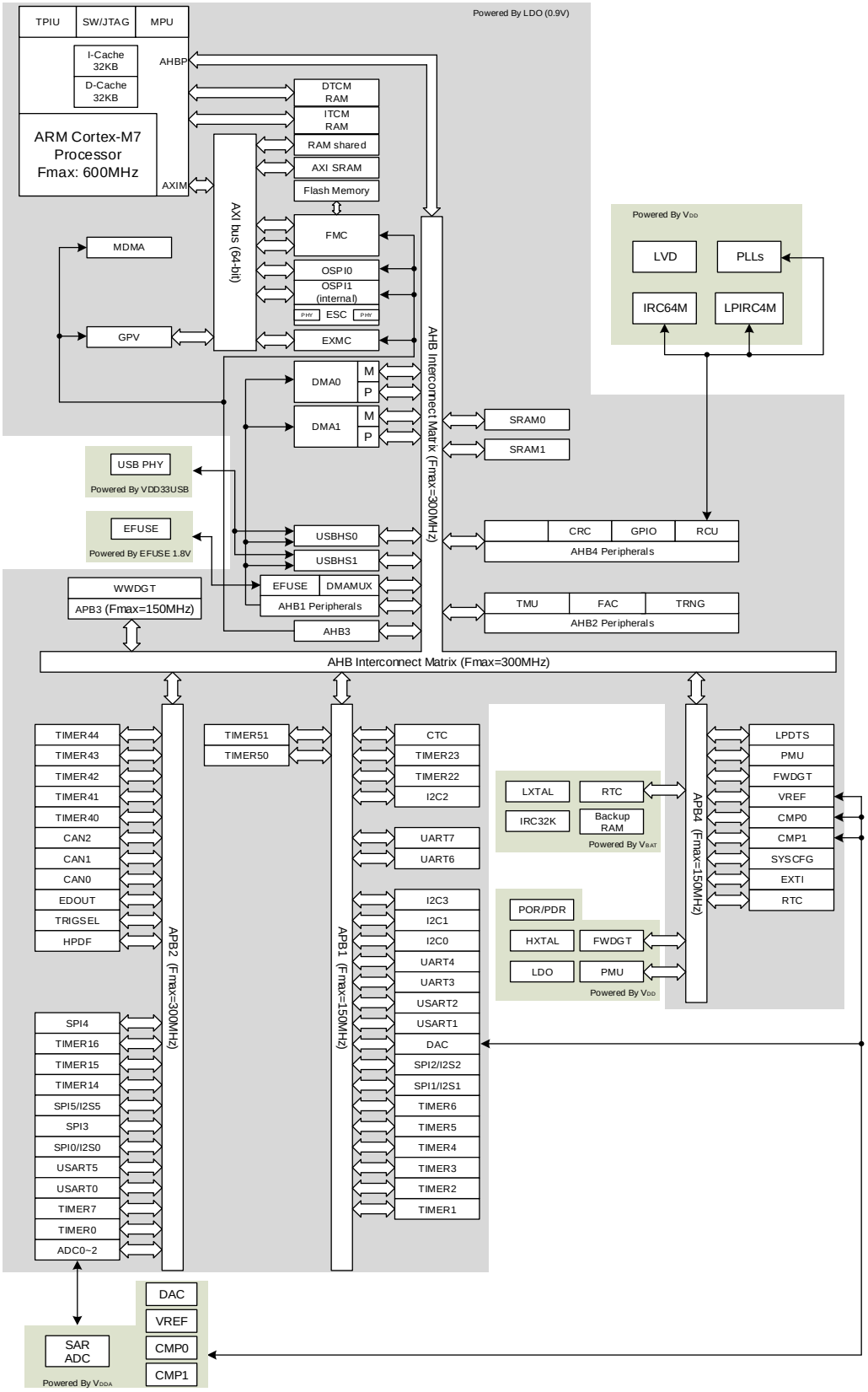


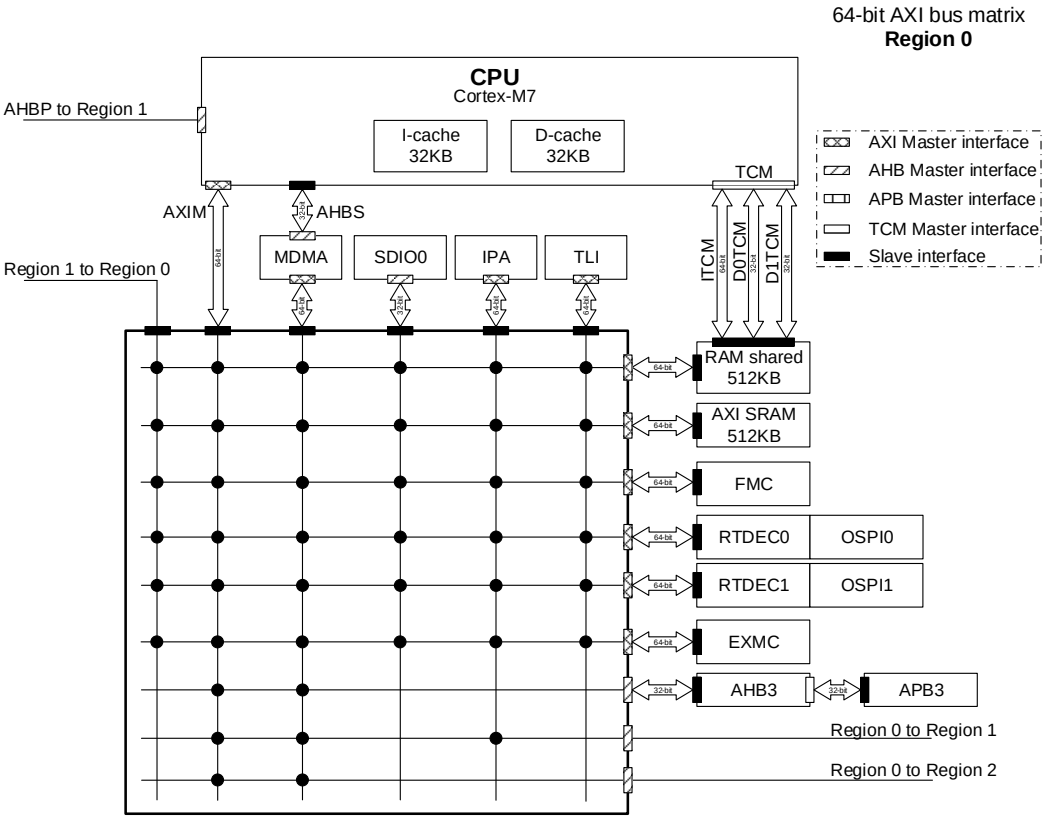
**Figure 1-3. The system architecture of GD32H73x_75x devices (For GD32H75E)**



### 1.2.1. Bus matrix Region 0

The 64-bit AXI bus matrix Region 0 is shown in ***Figure 1-4. Bus matrix Region 0***.

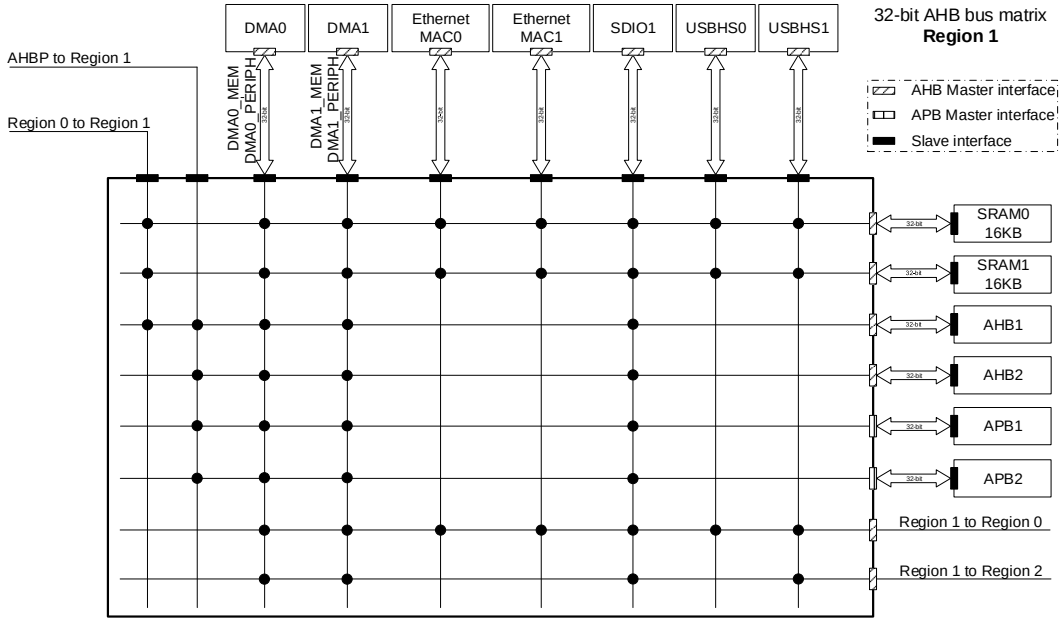
**Figure 1-4. Bus matrix Region 0**




### 1.2.2. Bus matrix Region 1

The 32-bit AHB bus matrix Region 1 is shown in ***Figure 1-5. Bus matrix Region 1***.

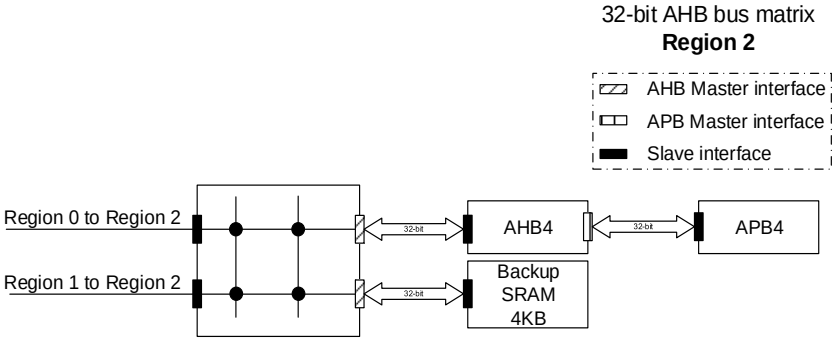
**Figure 1-5. Bus matrix Region 1**




## 1.2.3. Bus matrix Region 2

The 32-bit AHB bus matrix Region 2 is shown in ***Figure 1-6. Bus matrix Region 2***.

**Figure 1-6. Bus matrix Region 2**




As is shown above, there are 3 Regions of bus matrix, including I-Cache, D-Cache, TCM, MDMA, DMA0, DMA1, IPA, TLI, SDIO0, SDIO1, ENET, USBHS0 and USBHS1. The AXI bus matrix in Region 0 and AHB bus matrix in Region 1 and Region 2 provide guarantee and arbitration for concurrent access from multiple masters to multiple slaves. Arbitration adopts round-robin scheduling algorithm with QoS function in Region 0 and round-robin scheduling algorithm in Region 1 and Region 2. The ITCM and DTCM are directly connected to the Cortex®-M7 core through TCM buses and the accesses of ITCM and DTCM are zero-wait states. ENET is the Ethernet. TLI is the TFT LCD interface. USBHS is the high-speed USB. And IPA is the image processing accelerator.

## 1.3. Memory map

The Arm® Cortex®-M7 processor is structured in Harvard architecture which can use separate buses to fetch instructions and load/store data. ***Figure 1-2. The system architecture of GD32H73x_75x devices (For GD32H737_757_759)*** and ***Figure 1-3. The system architecture of GD32H73x_75x devices (For GD32H75E)*** show the memory map of the GD32H73x_75x series devices, including Code, SRAM, peripheral, and other pre-defined regions. Almost each peripheral is allocated 1KB of space. This allows simplifying the address decoding for each peripheral.

**Table 1-2. Memory map of GD32H73x_75x devices (For GD32H737_757_759)**

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| External RAM | | 0xD000 0000 - 0xDFFF FFFF | EXMC - SDRAM device 1 |
| | | 0xC000 0000 - 0xCFFF FFFF | EXMC - SDRAM device 0 (EXMC Bank 0 Region 0-3) |
| | | 0xA000 1000 - 0xBFFF FFFF | Reserved |
| | | 0xA000 0000 - 0xA000 0FFF | Reserved |
| | | 0x9000 0000 - 0x9FFF FFFF | OSPI0 |
| | | 0x8000 0000 - 0x8FFF FFFF | EXMC - NAND |
| | | 0x7000 0000 - 0x7FFF FFFF | OSPI1 |
| | | 0x6000 0000 - 0x6FFF FFFF | EXMC - NOR/PSRAM/SRAM |
| Peripheral | AHB4 | 0x5802 7000 - 0x5FFF FFFF | Reserved |
| | | 0x5802 6400 - 0x5802 67FF | HWSEM |
| | | 0x5802 6000 - 0x5802 63FF | Reserved |
| | | 0x5802 5000 - 0x5802 5FFF | Reserved |
| | | 0x5802 4C00 - 0x5802 4FFF | CRC |
| | | 0x5802 4800 - 0x5802 4BFF | Reserved |
| | | 0x5802 4400 - 0x5802 47FF | RCU |
| | | 0x5802 2C00 - 0x5802 43FF | Reserved |
| | | 0x5802 2800 - 0x5802 2BFF | GPIOK |
| | | 0x5802 2400 - 0x5802 27FF | GPIOJ |
| | | 0x5802 2000 - 0x5802 23FF | Reserved |
| | | 0x5802 1C00 - 0x5802 1FFF | GPIOH |
| | | 0x5802 1800 - 0x5802 1BFF | GPIOG |
| | | 0x5802 1400 - 0x5802 17FF | GPIOF |
| | | 0x5802 1000 - 0x5802 13FF | GPIOE |
| | | 0x5802 0C00 - 0x5802 0FFF | GPIOD |
| | | 0x5802 0800 - 0x5802 0BFF | GPIOC |
| | | 0x5802 0400 - 0x5802 07FF | GPIOB |
| | | 0x5802 0000 - 0x5802 03FF | GPIOA |
| | | 0x5801 0000 - 0x5801 FFFF | Reserved |
| | APB4 | 0x5800 7400 - 0x5800 FFFF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x5800 7000 - 0x5800 73FF | Reserved |
| | | 0x5800 6C00 - 0x5800 6FFF | Reserved |
| | | 0x5800 6800 - 0x5800 6BFF | LPDTS |
| | | 0x5800 5800 - 0x5800 67FF | PMU |
| | | 0x5800 5400 - 0x5800 57FF | Reserved |
| | | 0x5800 4C00 - 0x5800 53FF | Reserved |
| | | 0x5800 4800 - 0x5800 4BFF | FWDGT |
| | | 0x5800 4000 - 0x5800 43FF | RTC |
| | | 0x5800 3C00 - 0x5800 3FFF | VREF |
| | | 0x5800 3800 - 0x5800 3BFF | CMP0 - CMP1 |
| | | 0x5800 3400 - 0x5800 37FF | Reserved |
| | | 0x5800 3000 - 0x5800 33FF | Reserved |
| | | 0x5800 2C00 - 0x5800 2FFF | Reserved |
| | | 0x5800 2800 - 0x5800 2BFF | Reserved |
| | | 0x5800 2400 - 0x5800 27FF | Reserved |
| | | 0x5800 2000 - 0x5800 23FF | Reserved |
| | | 0x5800 1C00 - 0x5800 1FFF | Reserved |
| | | 0x5800 1400 - 0x5800 17FF | Reserved |
| | | 0x5800 0800 - 0x5800 13FF | Reserved |
| | | 0x5800 0400 - 0x5800 07FF | SYSCFG |
| | | 0x5800 0000 - 0x5800 03FF | EXTI |
| | AHB3 | 0x5200 C000 - 0x57FF FFFF | Reserved |
| | | 0x5200 BC00 - 0x5200 BFFF | RTDEC1 |
| | | 0x5200 B800 - 0x5200 BBFF | RTDEC0 |
| | | 0x5200 B400 - 0x5200 B7FF | OSPIM |
| | | 0x5200 B000 - 0x5200 B3FF | Reserved |
| | | 0x5200 A000 - 0x5200 AFFF | OSPI1 |
| | | 0x5200 9400 - 0x5200 9FFF | Reserved |
| | | 0x5200 9000 - 0x5200 93FF | RAMECCMU Region 0 |
| | | 0x5200 8000 - 0x5200 8FFF | CPDM(SDIO0) |
| | | 0x5200 7000 - 0x5200 7FFF | SDIO0 |
| | | 0x5200 6000 - 0x5200 6FFF | Reserved |
| | | 0x5200 5000 - 0x5200 5FFF | OSPI0 |
| | | 0x5200 4000 - 0x5200 4FFF | EXMC |
| | | 0x5200 3400 - 0x5200 3FFF | Reserved |
| | | 0x5200 3000 - 0x5200 33FF | Reserved |
| | | 0x5200 2000 - 0x5200 2FFF | Flash memory interface |
| | | 0x5200 1000 - 0x5200 1FFF | IPA |
| | | 0x5200 0000 - 0x5200 0FFF | MDMA |
| | | 0x5110 0000 - 0x51FF FFFF | Reserved |
| | | 0x5100 0000 - 0x510F FFFF | AXI interconnect matrix |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | APB3 | 0x5006 1000 - 0x50FF FFFF | Reserved |
| | | 0x5006 0C00 - 0x5006 0FFF | Reserved |
| | | 0x5006 0800 - 0x5006 0BFF | Reserved |
| | | 0x5006 0400 - 0x5006 07FF | Reserved |
| | | 0x5006 0000 - 0x5006 03FF | Reserved |
| | | 0x5005 0400 - 0x5005 FFFF | Reserved |
| | | 0x5005 0000 - 0x5005 03FF | Reserved |
| | | 0x5004 0000 - 0x5004 FFFF | Reserved |
| | | 0x5000 0000 - 0x5003 FFFF | Reserved |
| | | 0x5000 3000 - 0x5000 3FFF | WWDGT |
| | | 0x5000 2000 - 0x5000 2FFF | Reserved |
| | | 0x5000 1000 - 0x5000 1FFF | TLI |
| | | 0x5000 0000 - 0x5000 0FFF | Reserved |
| | AHB2 | 0x4802 5000 - 0x4FFF FFFF | Reserved(AHB2) |
| | | 0x4802 4800 - 0x4802 4FFF | FAC |
| | | 0x4802 4400 - 0x4802 47FF | TMU |
| | | 0x4802 4000 - 0x4802 43FF | Reserved |
| | | 0x4802 3000 - 0x4802 3FFF | RAMECCMU Region 1 |
| | | 0x4802 2C00 - 0x4802 2FFF | Reserved(AHB2) |
| | | 0x4802 2800 - 0x4802 2BFF | CPDM(SDIO1) |
| | | 0x4802 2400 - 0x4802 27FF | SDIO1 |
| | | 0x4802 1C00 - 0x4802 23FF | Reserved(AHB2) |
| | | 0x4802 1800 - 0x4802 1BFF | TRNG |
| | | 0x4802 1400 - 0x4802 17FF | HAU |
| | | 0x4802 1000 - 0x4802 13FF | CAU |
| | | 0x4802 0400 - 0x4802 0FFF | Reserved(AHB2) |
| | | 0x4802 0000 - 0x4802 03FF | DCI |
| | | 0x4800 1800 - 0x4801 FFFF | Reserved(AHB2) |
| | | 0x4800 1400 - 0x4800 17FF | Reserved |
| | | 0x4800 1000 - 0x4800 13FF | Reserved |
| | | 0x4800 0C00 - 0x4800 0FFF | Reserved |
| | | 0x4800 0800 - 0x4800 0BFF | Reserved |
| | | 0x4800 0400 - 0x4800 07FF | Reserved |
| | | 0x4800 0000 - 0x4800 03FF | Reserved |
| | AHB1 | 0x400C 0000 - 0x47FF FFFF | Reserved(AHB1) |
| | | 0x4008 0000 - 0x400B FFFF | USBHS1 |
| | | 0x4004 0000 - 0x4007 FFFF | USBHS0 |
| | | 0x4003 8C00 - 0x4003 FFFF | Reserved |
| | | 0x4003 8400 - 0x4003 8BFF | Reserved |
| | | 0x4003 8000 - 0x4003 83FF | Reserved |
| | | 0x4003 3000 - 0x4003 7FFF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4003 0000 - 0x4003 2FFF | Reserved |
| | | 0x4002 C000 - 0x4002 FFFF | Reserved |
| | | 0x4002 BC00 - 0x4002 BFFF | ENET1 |
| | | 0x4002 B000 - 0x4002 BBFF | |
| | | 0x4002 A000 - 0x4002 AFFF | |
| | | 0x4002 8000 - 0x4002 9FFF | ENET0 |
| | | 0x4002 6800 - 0x4002 7FFF | Reserved |
| | | 0x4002 6400 - 0x4002 67FF | Reserved |
| | | 0x4002 6000 - 0x4002 63FF | Reserved |
| | | 0x4002 5000 - 0x4002 5FFF | Reserved |
| | | 0x4002 4000 - 0x4002 4FFF | Reserved |
| | | 0x4002 3C00 - 0x4002 3FFF | Reserved |
| | | 0x4002 3800 - 0x4002 3BFF | Reserved |
| | | 0x4002 3400 - 0x4002 37FF | Reserved |
| | | 0x4002 3000 - 0x4002 33FF | Reserved |
| | | 0x4002 2C00 - 0x4002 2FFF | Reserved |
| | | 0x4002 2800 - 0x4002 2BFF | EFUSE |
| | | 0x4002 2400 - 0x4002 27FF | Reserved |
| | | 0x4002 2000 - 0x4002 23FF | Reserved |
| | | 0x4002 1C00 - 0x4002 1FFF | Reserved |
| | | 0x4002 1800 - 0x4002 1BFF | Reserved |
| | | 0x4002 1400 - 0x4002 17FF | Reserved |
| | | 0x4002 1000 - 0x4002 13FF | Reserved |
| | | 0x4002 0C00 - 0x4002 0FFF | Reserved |
| | | 0x4002 0800 - 0x4002 0BFF | DMAMUX |
| | | 0x4002 0400 - 0x4002 07FF | DMA1 |
| | | 0x4002 0000 - 0x4002 03FF | DMA0 |
| | APB2 | 0x4001 F400 - 0x4001 FFFF | Reserved |
| | | 0x4001 F000 - 0x4001 F3FF | TIMER44 |
| | | 0x4001 DC00 - 0x4001 DFFF | TIMER43 |
| | | 0x4001 D800 - 0x4001 DBFF | TIMER42 |
| | | 0x4001 D400 - 0x4001 D7FF | TIMER41 |
| | | 0x4001 D000 - 0x4001 D3FF | TIMER40 |
| | | 0x4001 C000 - 0x4001 CFFF | CAN2(4KB) |
| | | 0x4001 B000 - 0x4001 BFFF | CAN1(4KB) |
| | | 0x4001 A000 - 0x4001 AFFF | CAN0(4KB) |
| | | 0x4001 8C00 - 0x4001 9FFF | Reserved |
| | | 0x4001 8800 - 0x4001 8BFF | EDOUT |
| | | 0x4001 8400 - 0x4001 87FF | TRIGSEL |
| | | 0x4001 8000 - 0x4001 83FF | Reserved(APB2) |
| | | 0x4001 7C00 - 0x4001 7FFF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4001 7800 - 0x4001 7BFF | Reserved |
| | | 0x4001 7400 - 0x4001 77FF | Reserved |
| | | 0x4001 7000 - 0x4001 73FF | HPDF |
| | | 0x4001 6C00 - 0x4001 6FFF | Reserved |
| | | 0x4001 6800 - 0x4001 6BFF | Reserved |
| | | 0x4001 6400 - 0x4001 67FF | Reserved |
| | | 0x4001 6000 - 0x4001 63FF | SAI2 |
| | | 0x4001 5C00 - 0x4001 5FFF | SAI1 |
| | | 0x4001 5800 - 0x4001 5BFF | SAI0 |
| | | 0x4001 5400 - 0x4001 57FF | Reserved |
| | | 0x4001 5000 - 0x4001 53FF | SPI4 |
| | | 0x4001 4C00 - 0x4001 4FFF | Reserved |
| | | 0x4001 4800 - 0x4001 4BFF | TIMER16 |
| | | 0x4001 4400 - 0x4001 47FF | TIMER15 |
| | | 0x4001 4000 - 0x4001 43FF | TIMER14 |
| | | 0x4001 3C00 - 0x4001 3FFF | Reserved |
| | | 0x4001 3800 - 0x4001 3BFF | SPI5/I2S5 |
| | | 0x4001 3400 - 0x4001 37FF | SPI3 |
| | | 0x4001 3000 - 0x4001 33FF | SPI0/I2S0 |
| | | 0x4001 2C00 - 0x4001 2FFF | ADC2 |
| | | 0x4001 2800 - 0x4001 2BFF | ADC1 |
| | | 0x4001 2400 - 0x4001 27FF | ADC0 |
| | | 0x4001 2000 - 0x4001 23FF | Reserved |
| | | 0x4001 1C00 - 0x4001 1FFF | Reserved |
| | | 0x4001 1800 - 0x4001 1BFF | Reserved |
| | | 0x4001 1400 - 0x4001 17FF | USART5 |
| | | 0x4001 1000 - 0x4001 13FF | USART0 |
| | | 0x4001 0C00 - 0x4001 0FFF | Reserved |
| | | 0x4001 0800 - 0x4001 0BFF | Reserved |
| | | 0x4001 0400 - 0x4001 07FF | TIMER7 |
| | | 0x4001 0000 - 0x4001 03FF | TIMER0 |
| | APB1 | 0x4000 F800 - 0x4000 FFFF | Reserved |
| | | 0x4000 F400 - 0x4000 F7FF | TIMER51 |
| | | 0x4000 F000 - 0x4000 F3FF | TIMER50 |
| | | 0x4000 EC00 - 0x4000 EFFF | TIMER31 |
| | | 0x4000 E800 - 0x4000 EBFF | TIMER30 |
| | | 0x4000 E400 - 0x4000 E7FF | TIMER23 |
| | | 0x4000 E000 - 0x4000 E3FF | TIMER22 |
| | | 0x4000 DC00 - 0x4000 DFFF | Reserved |
| | | 0x4000 D800 - 0x4000 DBFF | Reserved |
| | | 0x4000 D400 - 0x4000 D7FF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4000 D000 - 0x4000 D3FF | Reserved |
| | | 0x4000 CC00 - 0x4000 CFFF | Reserved |
| | | 0x4000 C800 - 0x4000 CBFF | Reserved |
| | | 0x4000 C400 - 0x4000 C7FF | Reserved |
| | | 0x4000 C000 - 0x4000 C3FF | I2C2 |
| | | 0x4000 9800 - 0x4000 BFFF | Reserved |
| | | 0x4000 9400 - 0x4000 97FF | MDIO |
| | | 0x4000 8800 - 0x4000 93FF | Reserved |
| | | 0x4000 8400 - 0x4000 87FF | CTC |
| | | 0x4000 8000 - 0x4000 83FF | Reserved |
| | | 0x4000 7C00 - 0x4000 7FFF | UART7 |
| | | 0x4000 7800 - 0x4000 7BFF | UART6 |
| | | 0x4000 7400 - 0x4000 77FF | DAC |
| | | 0x4000 7000 - 0x4000 73FF | Reserved |
| | | 0x4000 6C00 - 0x4000 6FFF | Reserved |
| | | 0x4000 6800 - 0x4000 6BFF | Reserved |
| | | 0x4000 6400 - 0x4000 67FF | Reserved |
| | | 0x4000 6000 - 0x4000 63FF | Reserved |
| | | 0x4000 5C00 - 0x4000 5FFF | I2C3 |
| | | 0x4000 5800 - 0x4000 5BFF | I2C1 |
| | | 0x4000 5400 - 0x4000 57FF | I2C0 |
| | | 0x4000 5000 - 0x4000 53FF | UART4 |
| | | 0x4000 4C00 - 0x4000 4FFF | UART3 |
| | | 0x4000 4800 - 0x4000 4BFF | USART2 |
| | | 0x4000 4400 - 0x4000 47FF | USART1 |
| | | 0x4000 4000 - 0x4000 43FF | RSPDIF |
| | | 0x4000 3C00 - 0x4000 3FFF | SPI2/I2S2 |
| | | 0x4000 3800 - 0x4000 3BFF | SPI1/I2S1 |
| | | 0x4000 3400 - 0x4000 37FF | Reserved |
| | | 0x4000 3000 - 0x4000 33FF | Reserved |
| | | 0x4000 2C00 - 0x4000 2FFF | Reserved |
| | | 0x4000 2800 - 0x4000 2BFF | Reserved |
| | | 0x4000 2400 - 0x4000 27FF | Reserved |
| | | 0x4000 2000 - 0x4000 23FF | Reserved |
| | | 0x4000 1C00 - 0x4000 1FFF | Reserved |
| | | 0x4000 1800 - 0x4000 1BFF | Reserved |
| | | 0x4000 1400 - 0x4000 17FF | TIMER6 |
| | | 0x4000 1000 - 0x4000 13FF | TIMER5 |
| | | 0x4000 0C00 - 0x4000 0FFF | TIMER4 |
| | | 0x4000 0800 - 0x4000 0BFF | TIMER3 |
| | | 0x4000 0400 - 0x4000 07FF | TIMER2 |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4000 0000 - 0x4000 03FF | TIMER1 |
| SRAM | | 0x3880 1000 - 0x3FFF FFFF | Reserved |
| | | 0x3880 0000 - 0x3880 0FFF | Backup SRAM |
| | | 0x3000 8000 - 0x387F FFFF | Reserved |
| | | 0x3000 4000 - 0x3000 7FFF | SRAM1(16KB) |
| | | 0x3000 0000 - 0x3000 3FFF | SRAM0(16KB) |
| | | 0x2410 0000 - 0x2FFF FFFF | Reserved |
| | | 0x2408 0000 - 0x240F FFFF | RAM(512KB) shared (ITCM/DTCM/AXI) |
| | | 0x2400 0000 - 0x2407 FFFF | AXI SRAM(512KB) |
| | | 0x2008 0000 - 0x23FF FFFF | Reserved |
| | | 0x2007 0000 - 0x2007 FFFF | DTCM RAM(from RAM shared) |
| | | 0x2006 0000 - 0x2006 FFFF | |
| | | 0x2003 0000 - 0x2005 FFFF | |
| | | 0x2002 0000 - 0x2002 FFFF | |
| | | 0x2001 C000 - 0x2001 FFFF | |
| | | 0x2001 8000 - 0x2001 BFFF | |
| | | 0x2001 0000 - 0x2001 7FFF | |
| | | 0x2000 D000 - 0x2000 FFFF | |
| | | 0x2000 C000 - 0x2000 CFFF | |
| | | 0x2000 8000 - 0x2000 BFFF | |
| | | 0x2000 5000 - 0x2000 7FFF | |
| | | 0x2000 2000 - 0x2000 4FFF | |
| | | 0x2000 1000 - 0x2000 1FFF | |
| | | 0x2000 0000 - 0x2000 0FFF | |
| Code | | 0x1FFF FC10 - 0x1FFF FFFF | Reserved |
| | | 0x1FFF FC00 - 0x1FFF FC0F | Reserved |
| | | 0x1FFF F818 - 0x1FFF BFFF | Reserved |
| | | 0x1FFF F800 - 0x1FFF F817 | Reserved |
| | | 0x1FFF F000 - 0x1FFF F7FF | Reserved |
| | | 0x1FFF EC00 - 0x1FFF EFFF | Reserved |
| | | 0x1FFF C010 - 0x1FFF EBFF | Reserved |
| | | 0x1FFF C000 - 0x1FFF C00F | Reserved |
| | | 0x1FFF B000 - 0x1FFF BFFF | Reserved |
| | | 0x1FFF 8000 - 0x1FFF AFFF | Reserved |
| | | 0x1FFF 7A10 - 0x1FFF 7FFF | Reserved |
| | | 0x1FFF 7800 - 0x1FFF 7A0F | Reserved |
| | | 0x1FFF 7400 - 0x1FFF 77FF | Reserved |
| | | 0x1FFF 7000 - 0x1FFF 73FF | Reserved |
| | | 0x1FFF 0000 - 0x1FFF 6FFF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x1FFE C010 - 0x1FFE FFFF | Reserved |
| | | 0x1FFE C000 - 0x1FFE C00F | Reserved |
| | | 0x1FF6 0000 - 0x1FFE BFFF | Reserved |
| | | 0x1FF4 0000 - 0x1FF5 FFFF | Reserved |
| | | 0x1FF1 0000 - 0x1FF3 FFFF | Reserved |
| | | 0x1FF0 0000 - 0x1FF0 FFFF | System Memory |
| | | 0x1002 0000 - 0x1FEF FFFF | Reserved |
| | | 0x1001 0000 - 0x1001 FFFF | Reserved |
| | | 0x1000 0000 - 0x1000 FFFF | Reserved |
| | | 0x0A00 D000 - 0x0FFF FFFF | Reserved |
| | | 0x0A00 C000 - 0x0A00 CFFF | Reserved |
| | | 0x0A00 8000 - 0x0A00 BFFF | Reserved |
| | | 0x0A00 0000 - 0x0A00 7FFF | Reserved |
| | | 0x08C0 1000 - 0x09FF FFFF | Reserved |
| | | 0x08C0 0000 - 0x08C0 0FFF | Reserved |
| | | 0x0881 0000 - 0x08BF FFFF | Reserved |
| | | 0x0880 0000 - 0x0880 FFFF | Reserved |
| | | 0x0840 0000 - 0x087F FFFF | Reserved |
| | | 0x083C 0000 - 0x083F FFFF | Reserved |
| | | 0x0830 0000 - 0x083B FFFF | Flash memory |
| | | 0x0810 0000 - 0x082F FFFF | |
| | | 0x0808 0000 - 0x080F FFFF | |
| | | 0x0806 0000 - 0x0807 FFFF | |
| | | 0x0802 0000 - 0x0805 FFFF | |
| | | 0x0801 0000 - 0x0801 FFFF | |
| | | 0x0800 0000 - 0x0800 FFFF | |
| | | 0x0030 0000 - 0x07FF FFFF | Reserved |
| | | 0x0010 0000 - 0x002F FFFF | Reserved |
| | | 0x0008 0000 - 0x000F FFFF | Reserved |
| | | 0x0002 6000 - 0x0007 FFFF | ITCM RAM(from RAM shared) |
| | | 0x0002 0000 - 0x0002 5FFF | |
| | | 0x0001 0000 - 0x0001 FFFF | |
| | | 0x0000 0000 - 0x0000 FFFF | |

**Table 1-3. Memory map of GD32H73x_75x devices (For GD32H75E)**

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| External RAM | | 0xD000 0000 - 0xDFFF FFFF | EXMC - SDRAM device 1 |
| | | 0xC000 0000 - 0xCFFF FFFF | EXMC - SDRAM device 0 (EXMC Bank 0 Region 0-3) |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0xA000 1000 - 0xBFFF FFFF | Reserved |
| | | 0xA000 0000 - 0xA000 0FFF | Reserved |
| | | 0x9000 0000 - 0x9FFF FFFF | OSPI0 |
| | | 0x8000 0000 - 0x8FFF FFFF | EXMC - NAND |
| | | 0x7000 0000 - 0x7FFF FFFF | OSPI1 |
| | | 0x6000 0000 - 0x6FFF FFFF | EXMC - NOR/PSRAM/SRAM |
| Peripheral | AHB4 | 0x5802 7000 - 0x5FFF FFFF | Reserved |
| | | 0x5802 6400 - 0x5802 67FF | Reserved |
| | | 0x5802 6000 - 0x5802 63FF | Reserved |
| | | 0x5802 5000 - 0x5802 5FFF | Reserved |
| | | 0x5802 4C00 - 0x5802 4FFF | CRC |
| | | 0x5802 4800 - 0x5802 4BFF | Reserved |
| | | 0x5802 4400 - 0x5802 47FF | RCU |
| | | 0x5802 2C00 - 0x5802 43FF | Reserved |
| | | 0x5802 2800 - 0x5802 2BFF | Reserved |
| | | 0x5802 2400 - 0x5802 27FF | Reserved |
| | | 0x5802 2000 - 0x5802 23FF | Reserved |
| | | 0x5802 1C00 - 0x5802 1FFF | GPIOH |
| | | 0x5802 1800 - 0x5802 1BFF | GPIOG |
| | | 0x5802 1400 - 0x5802 17FF | GPIOF |
| | | 0x5802 1000 - 0x5802 13FF | GPIOE |
| | | 0x5802 0C00 - 0x5802 0FFF | GPIOD |
| | | 0x5802 0800 - 0x5802 0BFF | GPIOC |
| | | 0x5802 0400 - 0x5802 07FF | GPIOB |
| | | 0x5802 0000 - 0x5802 03FF | GPIOA |
| | | 0x5801 0000 - 0x5801 FFFF | Reserved |
| | APB4 | 0x5800 7400 - 0x5800 FFFF | Reserved |
| | | 0x5800 7000 - 0x5800 73FF | Reserved |
| | | 0x5800 6C00 - 0x5800 6FFF | Reserved |
| | | 0x5800 6800 - 0x5800 6BFF | LPDTS |
| | | 0x5800 5800 - 0x5800 67FF | PMU |
| | | 0x5800 5400 - 0x5800 57FF | Reserved |
| | | 0x5800 4C00 - 0x5800 53FF | Reserved |
| | | 0x5800 4800 - 0x5800 4BFF | FWDGT |
| | | 0x5800 4000 - 0x5800 43FF | RTC |
| | | 0x5800 3C00 - 0x5800 3FFF | VREF |
| | | 0x5800 3800 - 0x5800 3BFF | CMP0 - CMP1 |
| | | 0x5800 3400 - 0x5800 37FF | Reserved |
| | | 0x5800 3000 - 0x5800 33FF | Reserved |
| | | 0x5800 2C00 - 0x5800 2FFF | Reserved |
| | | 0x5800 2800 - 0x5800 2BFF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x5800 2400 - 0x5800 27FF | Reserved |
| | | 0x5800 2000 - 0x5800 23FF | Reserved |
| | | 0x5800 1C00 - 0x5800 1FFF | Reserved |
| | | 0x5800 1400 - 0x5800 17FF | Reserved |
| | | 0x5800 0800 - 0x5800 13FF | Reserved |
| | | 0x5800 0400 - 0x5800 07FF | SYSCFG |
| | | 0x5800 0000 - 0x5800 03FF | EXTI |
| | AHB3 | 0x5200 C000 - 0x57FF FFFF | Reserved |
| | | 0x5200 BC00 - 0x5200 BFFF | Reserved |
| | | 0x5200 B800 - 0x5200 BBFF | Reserved |
| | | 0x5200 B400 - 0x5200 B7FF | OSPIM |
| | | 0x5200 B000 - 0x5200 B3FF | Reserved |
| | | 0x5200 A000 - 0x5200 AFFF | OSPI1(internal) |
| | | 0x5200 9400 - 0x5200 9FFF | Reserved |
| | | 0x5200 9000 - 0x5200 93FF | RAMECCMU Region 0 |
| | | 0x5200 8000 - 0x5200 8FFF | Reserved |
| | | 0x5200 7000 - 0x5200 7FFF | Reserved |
| | | 0x5200 6000 - 0x5200 6FFF | Reserved |
| | | 0x5200 5000 - 0x5200 5FFF | OSPI0 |
| | | 0x5200 4000 - 0x5200 4FFF | EXMC |
| | | 0x5200 3400 - 0x5200 3FFF | Reserved |
| | | 0x5200 3000 - 0x5200 33FF | Reserved |
| | | 0x5200 2000 - 0x5200 2FFF | Flash memory interface |
| | | 0x5200 1000 - 0x5200 1FFF | Reserved |
| | | 0x5200 0000 - 0x5200 0FFF | MDMA |
| | | 0x5110 0000 - 0x51FF FFFF | Reserved |
| | | 0x5100 0000 - 0x510F FFFF | AXI interconnect matrix |
| | APB3 | 0x5006 1000 - 0x50FF FFFF | Reserved |
| | | 0x5006 0C00 - 0x5006 0FFF | Reserved |
| | | 0x5006 0800 - 0x5006 0BFF | Reserved |
| | | 0x5006 0400 - 0x5006 07FF | Reserved |
| | | 0x5006 0000 - 0x5006 03FF | Reserved |
| | | 0x5005 0400 - 0x5005 FFFF | Reserved |
| | | 0x5005 0000 - 0x5005 03FF | Reserved |
| | | 0x5004 0000 - 0x5004 FFFF | Reserved |
| | | 0x5000 0000 - 0x5003 FFFF | Reserved |
| | | 0x5000 3000 - 0x5000 3FFF | WWDGT |
| | | 0x5000 2000 - 0x5000 2FFF | Reserved |
| | | 0x5000 1000 - 0x5000 1FFF | Reserved |
| | | 0x5000 0000 - 0x5000 0FFF | Reserved |
| | AHB2 | 0x4802 5000 - 0x4FFF FFFF | Reserved(AHB2) |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4802 4800 - 0x4802 4FFF | FAC |
| | | 0x4802 4400 - 0x4802 47FF | TMU |
| | | 0x4802 4000 - 0x4802 43FF | Reserved |
| | | 0x4802 3000 - 0x4802 3FFF | RAMECCMU Region 1 |
| | | 0x4802 2C00 - 0x4802 2FFF | Reserved(AHB2) |
| | | 0x4802 2800 - 0x4802 2BFF | Reserved |
| | | 0x4802 2400 - 0x4802 27FF | Reserved |
| | | 0x4802 1C00 - 0x4802 23FF | Reserved(AHB2) |
| | | 0x4802 1800 - 0x4802 1BFF | TRNG |
| | | 0x4802 1400 - 0x4802 17FF | Reserved |
| | | 0x4802 1000 - 0x4802 13FF | Reserved |
| | | 0x4802 0400 - 0x4802 0FFF | Reserved(AHB2) |
| | | 0x4802 0000 - 0x4802 03FF | Reserved |
| | | 0x4800 1800 - 0x4801 FFFF | Reserved(AHB2) |
| | | 0x4800 1400 - 0x4800 17FF | Reserved |
| | | 0x4800 1000 - 0x4800 13FF | Reserved |
| | | 0x4800 0C00 - 0x4800 0FFF | Reserved |
| | | 0x4800 0800 - 0x4800 0BFF | Reserved |
| | | 0x4800 0400 - 0x4800 07FF | Reserved |
| | | 0x4800 0000 - 0x4800 03FF | Reserved |
| | AHB1 | 0x400C 0000 - 0x47FF FFFF | Reserved(AHB1) |
| | | 0x4008 0000 - 0x400B FFFF | USBHS1 |
| | | 0x4004 0000 - 0x4007 FFFF | USBHS0 |
| | | 0x4003 8C00 - 0x4003 FFFF | Reserved |
| | | 0x4003 8400 - 0x4003 8BFF | Reserved |
| | | 0x4003 8000 - 0x4003 83FF | Reserved |
| | | 0x4003 3000 - 0x4003 7FFF | Reserved |
| | | 0x4003 0000 - 0x4003 2FFF | Reserved |
| | | 0x4002 C000 - 0x4002 FFFF | Reserved |
| | | 0x4002 BC00 - 0x4002 BFFF | Reserved |
| | | 0x4002 B000 - 0x4002 BBFF | |
| | | 0x4002 A000 - 0x4002 AFFF | |
| | | 0x4002 8000 - 0x4002 9FFF | Reserved |
| | | 0x4002 6800 - 0x4002 7FFF | Reserved |
| | | 0x4002 6400 - 0x4002 67FF | Reserved |
| | | 0x4002 6000 - 0x4002 63FF | Reserved |
| | | 0x4002 5000 - 0x4002 5FFF | Reserved |
| | | 0x4002 4000 - 0x4002 4FFF | Reserved |
| | | 0x4002 3C00 - 0x4002 3FFF | Reserved |
| | | 0x4002 3800 - 0x4002 3BFF | Reserved |
| | | 0x4002 3400 - 0x4002 37FF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4002 3000 - 0x4002 33FF | Reserved |
| | | 0x4002 2C00 - 0x4002 2FFF | Reserved |
| | | 0x4002 2800 - 0x4002 2BFF | EFUSE |
| | | 0x4002 2400 - 0x4002 27FF | Reserved |
| | | 0x4002 2000 - 0x4002 23FF | Reserved |
| | | 0x4002 1C00 - 0x4002 1FFF | Reserved |
| | | 0x4002 1800 - 0x4002 1BFF | Reserved |
| | | 0x4002 1400 - 0x4002 17FF | Reserved |
| | | 0x4002 1000 - 0x4002 13FF | Reserved |
| | | 0x4002 0C00 - 0x4002 0FFF | Reserved |
| | | 0x4002 0800 - 0x4002 0BFF | DMAMUX |
| | | 0x4002 0400 - 0x4002 07FF | DMA1 |
| | | 0x4002 0000 - 0x4002 03FF | DMA0 |
| | APB2 | 0x4001 F400 - 0x4001 FFFF | Reserved |
| | | 0x4001 F000 - 0x4001 F3FF | TIMER44 |
| | | 0x4001 DC00 - 0x4001 DFFF | TIMER43 |
| | | 0x4001 D800 - 0x4001 DBFF | TIMER42 |
| | | 0x4001 D400 - 0x4001 D7FF | TIMER41 |
| | | 0x4001 D000 - 0x4001 D3FF | TIMER40 |
| | | 0x4001 C000 - 0x4001 CFFF | CAN2(4KB) |
| | | 0x4001 B000 - 0x4001 BFFF | CAN1(4KB) |
| | | 0x4001 A000 - 0x4001 AFFF | CAN0(4KB) |
| | | 0x4001 8C00 - 0x4001 9FFF | Reserved |
| | | 0x4001 8800 - 0x4001 8BFF | EDOUT |
| | | 0x4001 8400 - 0x4001 87FF | TRIGSEL |
| | | 0x4001 8000 - 0x4001 83FF | Reserved(APB2) |
| | | 0x4001 7C00 - 0x4001 7FFF | Reserved |
| | | 0x4001 7800 - 0x4001 7BFF | Reserved |
| | | 0x4001 7400 - 0x4001 77FF | Reserved |
| | | 0x4001 7000 - 0x4001 73FF | HPDF |
| | | 0x4001 6C00 - 0x4001 6FFF | Reserved |
| | | 0x4001 6800 - 0x4001 6BFF | Reserved |
| | | 0x4001 6400 - 0x4001 67FF | Reserved |
| | | 0x4001 6000 - 0x4001 63FF | Reserved |
| | | 0x4001 5C00 - 0x4001 5FFF | Reserved |
| | | 0x4001 5800 - 0x4001 5BFF | Reserved |
| | | 0x4001 5400 - 0x4001 57FF | Reserved |
| | | 0x4001 5000 - 0x4001 53FF | SPI4 |
| | | 0x4001 4C00 - 0x4001 4FFF | Reserved |
| | | 0x4001 4800 - 0x4001 4BFF | TIMER16 |
| | | 0x4001 4400 - 0x4001 47FF | TIMER15 |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4001 4000 - 0x4001 43FF | TIMER14 |
| | | 0x4001 3C00 - 0x4001 3FFF | Reserved |
| | | 0x4001 3800 - 0x4001 3BFF | SPI5/I2S5 |
| | | 0x4001 3400 - 0x4001 37FF | SPI3 |
| | | 0x4001 3000 - 0x4001 33FF | SPI0/I2S0 |
| | | 0x4001 2C00 - 0x4001 2FFF | ADC2 |
| | | 0x4001 2800 - 0x4001 2BFF | ADC1 |
| | | 0x4001 2400 - 0x4001 27FF | ADC0 |
| | | 0x4001 2000 - 0x4001 23FF | Reserved |
| | | 0x4001 1C00 - 0x4001 1FFF | Reserved |
| | | 0x4001 1800 - 0x4001 1BFF | Reserved |
| | | 0x4001 1400 - 0x4001 17FF | USART5 |
| | | 0x4001 1000 - 0x4001 13FF | USART0 |
| | | 0x4001 0C00 - 0x4001 0FFF | Reserved |
| | | 0x4001 0800 - 0x4001 0BFF | Reserved |
| | | 0x4001 0400 - 0x4001 07FF | TIMER7 |
| | | 0x4001 0000 - 0x4001 03FF | TIMER0 |
| | APB1 | 0x4000 F800 - 0x4000 FFFF | Reserved |
| | | 0x4000 F400 - 0x4000 F7FF | TIMER51 |
| | | 0x4000 F000 - 0x4000 F3FF | TIMER50 |
| | | 0x4000 EC00 - 0x4000 EFFF | Reserved |
| | | 0x4000 E800 - 0x4000 EBFF | Reserved |
| | | 0x4000 E400 - 0x4000 E7FF | TIMER23 |
| | | 0x4000 E000 - 0x4000 E3FF | TIMER22 |
| | | 0x4000 DC00 - 0x4000 DFFF | Reserved |
| | | 0x4000 D800 - 0x4000 DBFF | Reserved |
| | | 0x4000 D400 - 0x4000 D7FF | Reserved |
| | | 0x4000 D000 - 0x4000 D3FF | Reserved |
| | | 0x4000 CC00 - 0x4000 CFFF | Reserved |
| | | 0x4000 C800 - 0x4000 CBFF | Reserved |
| | | 0x4000 C400 - 0x4000 C7FF | Reserved |
| | | 0x4000 C000 - 0x4000 C3FF | I2C2 |
| | | 0x4000 9800 - 0x4000 BFFF | Reserved |
| | | 0x4000 9400 - 0x4000 97FF | Reserved |
| | | 0x4000 8800 - 0x4000 93FF | Reserved |
| | | 0x4000 8400 - 0x4000 87FF | CTC |
| | | 0x4000 8000 - 0x4000 83FF | Reserved |
| | | 0x4000 7C00 - 0x4000 7FFF | UART7 |
| | | 0x4000 7800 - 0x4000 7BFF | UART6 |
| | | 0x4000 7400 - 0x4000 77FF | DAC |
| | | 0x4000 7000 - 0x4000 73FF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x4000 6C00 - 0x4000 6FFF | Reserved |
| | | 0x4000 6800 - 0x4000 6BFF | Reserved |
| | | 0x4000 6400 - 0x4000 67FF | Reserved |
| | | 0x4000 6000 - 0x4000 63FF | Reserved |
| | | 0x4000 5C00 - 0x4000 5FFF | I2C3 |
| | | 0x4000 5800 - 0x4000 5BFF | I2C1 |
| | | 0x4000 5400 - 0x4000 57FF | I2C0 |
| | | 0x4000 5000 - 0x4000 53FF | UART4 |
| | | 0x4000 4C00 - 0x4000 4FFF | UART3 |
| | | 0x4000 4800 - 0x4000 4BFF | USART2 |
| | | 0x4000 4400 - 0x4000 47FF | USART1 |
| | | 0x4000 4000 - 0x4000 43FF | Reserved |
| | | 0x4000 3C00 - 0x4000 3FFF | SPI2/I2S2 |
| | | 0x4000 3800 - 0x4000 3BFF | SPI1/I2S1 |
| | | 0x4000 3400 - 0x4000 37FF | Reserved |
| | | 0x4000 3000 - 0x4000 33FF | Reserved |
| | | 0x4000 2C00 - 0x4000 2FFF | Reserved |
| | | 0x4000 2800 - 0x4000 2BFF | Reserved |
| | | 0x4000 2400 - 0x4000 27FF | Reserved |
| | | 0x4000 2000 - 0x4000 23FF | Reserved |
| | | 0x4000 1C00 - 0x4000 1FFF | Reserved |
| | | 0x4000 1800 - 0x4000 1BFF | Reserved |
| | | 0x4000 1400 - 0x4000 17FF | TIMER6 |
| | | 0x4000 1000 - 0x4000 13FF | TIMER5 |
| | | 0x4000 0C00 - 0x4000 0FFF | TIMER4 |
| | | 0x4000 0800 - 0x4000 0BFF | TIMER3 |
| | | 0x4000 0400 - 0x4000 07FF | TIMER2 |
| | | 0x4000 0000 - 0x4000 03FF | TIMER1 |
| SRAM | | 0x3880 1000 - 0x3FFF FFFF | Reserved |
| | | 0x3880 0000 - 0x3880 0FFF | Backup SRAM |
| | | 0x3000 8000 - 0x387F FFFF | Reserved |
| | | 0x3000 4000 - 0x3000 7FFF | SRAM1(16KB) |
| | | 0x3000 0000 - 0x3000 3FFF | SRAM0(16KB) |
| | | 0x2410 0000 - 0x2FFF FFFF | Reserved |
| | | 0x2408 0000 - 0x240F FFFF | RAM(512KB) shared (ITCM/DTCM/AXI) |
| | | 0x2400 0000 - 0x2407 FFFF | AXI SRAM(512KB) |
| | | 0x2008 0000 - 0x23FF FFFF | Reserved |
| | | 0x2007 0000 - 0x2007 FFFF | DTCM RAM(from RAM shared) |
| | | 0x2006 0000 - 0x2006 FFFF | |
| | | 0x2003 0000 - 0x2005 FFFF | |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x2002 0000 - 0x2002 FFFF | |
| | | 0x2001 C000 - 0x2001 FFFF | |
| | | 0x2001 8000 - 0x2001 BFFF | |
| | | 0x2001 0000 - 0x2001 7FFF | |
| | | 0x2000 D000 - 0x2000 FFFF | |
| | | 0x2000 C000 - 0x2000 CFFF | |
| | | 0x2000 8000 - 0x2000 BFFF | |
| | | 0x2000 5000 - 0x2000 7FFF | |
| | | 0x2000 2000 - 0x2000 4FFF | |
| | | 0x2000 1000 - 0x2000 1FFF | |
| | | 0x2000 0000 - 0x2000 0FFF | |
| Code | | 0x1FFF FC10 - 0x1FFF FFFF | Reserved |
| | | 0x1FFF FC00 - 0x1FFF FC0F | Reserved |
| | | 0x1FFF F818 - 0x1FFF BFFF | Reserved |
| | | 0x1FFF F800 - 0x1FFF F817 | Reserved |
| | | 0x1FFF F000 - 0x1FFF F7FF | Reserved |
| | | 0x1FFF EC00 - 0x1FFF EFFF | Reserved |
| | | 0x1FFF C010 - 0x1FFF EBFF | Reserved |
| | | 0x1FFF C000 - 0x1FFF C00F | Reserved |
| | | 0x1FFF B000 - 0x1FFF BFFF | Reserved |
| | | 0x1FFF 8000 - 0x1FFF AFFF | Reserved |
| | | 0x1FFF 7A10 - 0x1FFF 7FFF | Reserved |
| | | 0x1FFF 7800 - 0x1FFF 7A0F | Reserved |
| | | 0x1FFF 7400 - 0x1FFF 77FF | Reserved |
| | | 0x1FFF 7000 - 0x1FFF 73FF | Reserved |
| | | 0x1FFF 0000 - 0x1FFF 6FFF | Reserved |
| | | 0x1FFE C010 - 0x1FFE FFFF | Reserved |
| | | 0x1FFE C000 - 0x1FFE C00F | Reserved |
| | | 0x1FF6 0000 - 0x1FFE BFFF | Reserved |
| | | 0x1FF4 0000 - 0x1FF5 FFFF | Reserved |
| | | 0x1FF1 0000 - 0x1FF3 FFFF | Reserved |
| | | 0x1FF0 0000 - 0x1FF0 FFFF | System Memory |
| | | 0x1002 0000 - 0x1FEF FFFF | Reserved |
| | | 0x1001 0000 - 0x1001 FFFF | Reserved |
| | | 0x1000 0000 - 0x1000 FFFF | Reserved |
| | | 0x0A00 D000 - 0x0FFF FFFF | Reserved |
| | | 0x0A00 C000 - 0x0A00 CFFF | Reserved |
| | | 0x0A00 8000 - 0x0A00 BFFF | Reserved |
| | | 0x0A00 0000 - 0x0A00 7FFF | Reserved |
| | | 0x08C0 1000 - 0x09FF FFFF | Reserved |

| Pre-defined Regions | Bus | Address | Peripherals |
|---|---|---|---|
| | | 0x08C0 0000 - 0x08C0 0FFF | Reserved |
| | | 0x0881 0000 - 0x08BF FFFF | Reserved |
| | | 0x0880 0000 - 0x0880 FFFF | Reserved |
| | | 0x0840 0000 - 0x087F FFFF | Reserved |
| | | 0x083C 0000 - 0x083F FFFF | Reserved |
| | | 0x0830 0000 - 0x083B FFFF | Flash memory |
| | | 0x0810 0000 - 0x082F FFFF | |
| | | 0x0808 0000 - 0x080F FFFF | |
| | | 0x0806 0000 - 0x0807 FFFF | |
| | | 0x0802 0000 - 0x0805 FFFF | |
| | | 0x0801 0000 - 0x0801 FFFF | |
| | | 0x0800 0000 - 0x0800 FFFF | |
| | | 0x0030 0000 - 0x07FF FFFF | Reserved |
| | | 0x0010 0000 - 0x002F FFFF | Reserved |
| | | 0x0008 0000 - 0x000F FFFF | Reserved |
| | | 0x0002 6000 - 0x0007 FFFF | ITCM RAM(from RAM shared) |
| | | 0x0002 0000 - 0x0002 5FFF | |
| | | 0x0001 0000 - 0x0001 FFFF | |
| | | 0x0000 0000 - 0x0000 FFFF | |

### 1.3.1. On-chip SRAM memory

The devices of GD32H73x_75x series contain up to 512KB of on-chip SRAM (AXI SRAM), 4KB of backup SRAM and up to 512KB RAM shared by ITCM/DTCM/AXI SRAM. All of AHB SRAM support byte, half-word (16 bits), and word (32 bits) accesses. The on-chip SRAM (AXI SRAM) support byte, half-word (16 bits), word (32 bits) and double words (64 bits) accesses. SRAM0 and SRAM1 can be accessed by almost all AHB masters. The backup SRAM (BKPSRAM) is implemented in the backup domain, which can keep its content even when the $V_{DD}$ power supply is down.

**ITCM/DTCM SRAM access frequency**

If TCM_WAITSTATE in SYSCFG_SRAMCFG1 register is set to 0, the corresponding TCM SRAM can be accessed below a maximum frequency of $f_{TWW}$ which can be referred to the datasheet. TCM_WAITSTATE is set to 0 after system reset.

**ITCM/DTCM SRAM ECC function**

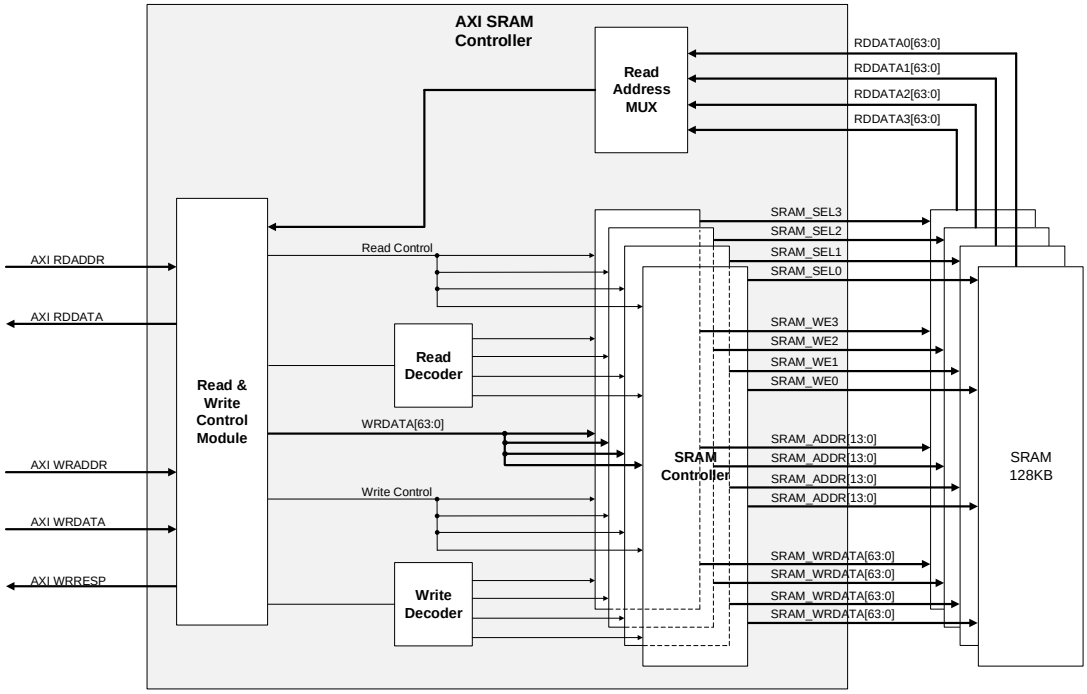
The TCM ECC function (ITCMECCEN / DTCM0ECCEN / DTCM1ECCEN) can be enabled to increase the robustness.

**AXI SRAM**

The on-chip SRAM (AXI SRAM) controller is consisted of read and write control module, MUX,

decoder and SRAM controllers. There is an arbiter with round-robin scheme for each SRAM controller. The block digram is shown in ***Figure 1-7. Block digram of AXI SRAM controller***.

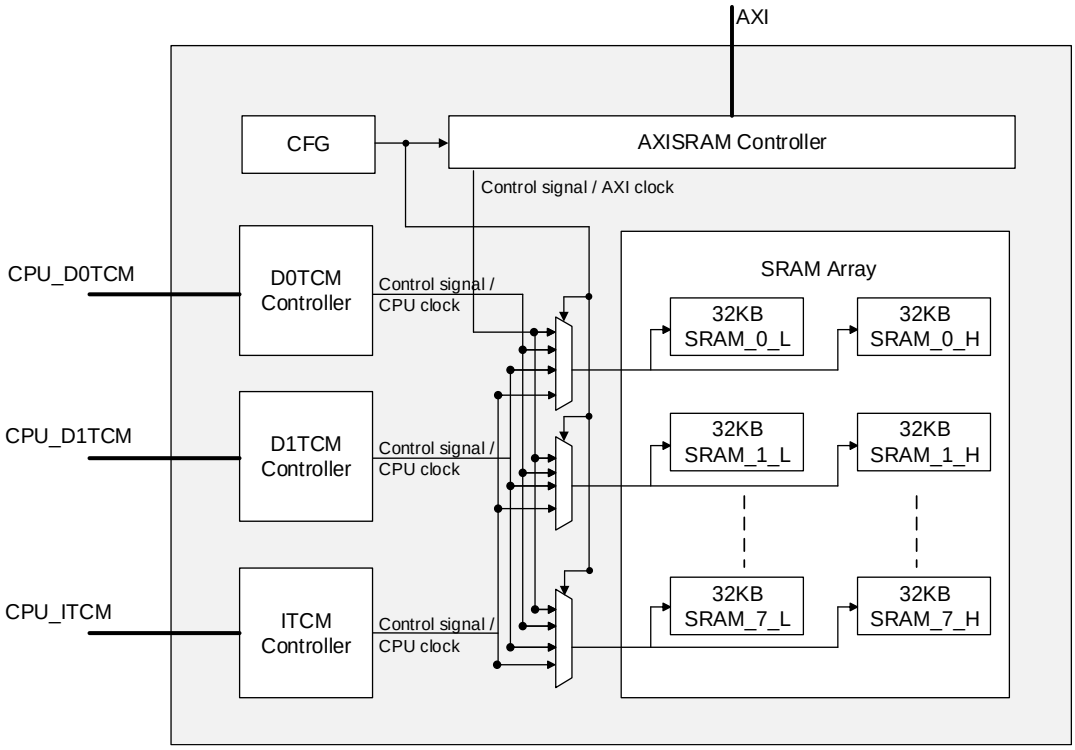
**Figure 1-7. Block digram of AXI SRAM controller**




**RAM shared by ITCM/DTCM/AXI SRAM**

512KB of RAM can be used by ITCM or DTCM or AXI SRAM, which can be configured through ITCM_SZ_SHRRAM and DTCM_SZ_SHRRAM bits in option byte status register 1 register, as described in ***Table 1-4. Configuration of ITCM/DTCM/AXI SRAM***. And the block digram is shown in ***Figure 1-8. Block digram of RAM shared by ITCM/DTCM/AXI SRAM***.

**Figure 1-8. Block digram of RAM shared by ITCM/DTCM/AXI SRAM**




The total capacity of ITCM, DTCM, and AXI SRAM is 512KB, of which ITCM configuration takes priority, DTCM configuration takes the second place, and the rest is the shared SRAM.

**Table 1-4. Configuration of ITCM/DTCM/AXI SRAM**

| ITCM_SZ_SHRRAM[3:0] | DTCM_SZ_SHRRAM[3:0] | ITCM size(KB) | DTCM size(KB) | AXI SRAM size(KB) |
|---|---|---|---|---|
| 0000 | 0000 | 0 | 0 | 512 |
| 0000 | 0111 | 0 | 64 | 448 |
| 0000 | 1000 | 0 | 128 | 384 |
| 0000 | 1001 | 0 | 256 | 256 |
| 0000 | 1010 | 0 | 512 | 0 |
| 0111 | 0000 | 64 | 0 | 448 |
| 0111 | 0111 | 64 | 64 | 384 |
| 0111 | 1000 | 64 | 128 | 320 |
| 0111 | 1001 | 64 | 256 | 192 |
| 1000 | 0000 | 128 | 0 | 384 |
| 1000 | 0111 | 128 | 64 | 320 |
| 1000 | 1000 | 128 | 128 | 256 |
| 1000 | 1001 | 128 | 256 | 128 |
| 1001 | 0000 | 256 | 0 | 256 |
| 1001 | 0111 | 256 | 64 | 192 |
| 1001 | 1000 | 256 | 128 | 128 |
| 1001 | 1001 | 256 | 256 | 0 |
| 1010 | 0000 | 512 | 0 | 0 |

### 1.3.2. On-chip flash memory overview

The devices provide high density on-chip flash memory, which is organized as follows:

- Up to 3840KB of main flash memory
- Up to 64KB of information blocks for the boot loader
- Option bytes to configure the device

Refer to ***Flash Memory Controller (FMC)*** Chapter for more details.

## 1.4. Boot configuration

The GD32H73x_75x devices provide different boot sources which can be selected by the BOOT pin and BOOT_ADDR0/1[15:0] in Boot address for Arm® Cortex®-M7 core register (FMC_BTADDR_MDF). The details are shown in ***Table 1-5. Boot mode selection***, ***Table 1-6. Details of Boot mode (For GD32H737_757_759)*** and ***Table 1-7. Details of Boot mode (For GD32H75E)***.

The value on the BOOT pin is latched on the 4th rising edge of CK_SYS after a reset. It is up to the user to set the BOOT pin after a power-on reset or a system reset to select the required boot source. Once the BOOT pin has been sampled, it is free and can be used for other purposes. The BOOT_ADDR0[15:0] and BOOT_ADDR1[15:0] address allows to configure the boot memory address to any address from 0x0000 0000 to 0x9000 0000. The boot mode can be obtained from the BOOT_MODE[2:0] bits in the SYSCFG_USERCFG register.

**Table 1-5. Boot mode selection**

| Boot source address | Boot mode selection pin |
|---|---|
| | BOOT |
| MSB of the boot address: defined by BOOT_ADDR0[15:0]<br>LSB of the boot address: 0x0000 | 0 |
| MSB of the boot address: defined by BOOT_ADDR1[15:0]<br>LSB of the boot address: 0x0000 | 1 |

**Table 1-6. Details of Boot mode (For GD32H737_757_759)**

| SCR | SPC[7:0] | BOOT_ADDRESS (configured in BOOT_ADDRx(x = 0,1)) | BOOT_MODE | Boot from |
|---|---|---|---|---|
| 1 | x | XXXX | SECURITY BOOT | ROM |
| 0 | Protection level high | 0x0800_0000~max user flash | USER BOOT | BOOT_ADDRESS |
| | | other | USER BOOT | 0x0800_0000 |
| | Protection level low | 0x2408_0000~ max RAM shared(ITCM/DTCM/AXI) | SRAM BOOT(RAM shared) | BOOT_ADDRESS |

| SCR | SPC[7:0] | BOOT_ADDRESS (configured in BOOT_ADDRx(x = 0,1)) | BOOT_MODE | Boot from |
|---|---|---|---|---|
| | | 0x2400_0000~ max AXI SRAM | SRAM BOOT(AXI SRAM) | BOOT_ADDRESS |
| | | 0x2000_0000 | SRAM BOOT(DTCM) | 0x2000_0000 |
| | | 0x0800_0000~max user flash | USER BOOT | BOOT_ADDRESS |
| | | 0x0000_0000 | SRAM BOOT(ITCM) | 0x0000_0000 |
| | | 0x1FF0_0000 | SYSTEM BOOT | BootLoader |
| | | Other | USER BOOT | 0x0800_0000(BOOT Pin = 0) |
| | | | SYSTEM BOOT | BootLoader(BOOT Pin = 1) |
| | No protection | 0x9000_0000 | USER BOOT | OSPI0 |
| | | 0x7000_0000 | USER BOOT | OSPI1 |
| | | 0x2408_0000~ max RAM shared(ITCM/DTCM/AXI) | SRAM BOOT(RAM shared) | BOOT_ADDRESS |
| | | 0x2400_0000~ max AXI SRAM | SRAM BOOT(AXI SRAM) | BOOT_ADDRESS |
| | | 0x2000_0000 | SRAM BOOT(DTCM) | 0x2000_0000 |
| | | 0x0800_0000~max user flash | USER BOOT | BOOT_ADDRESS |
| | | 0x0000_0000 | SRAM BOOT(ITCM) | 0x0000_0000 |
| | | 0x1FF0_0000 | SYSTEM BOOT | BootLoader |
| | | Other | USER BOOT | 0x0800_0000(BOOT Pin = 0) |
| | | | SYSTEM BOOT | BootLoader(BOOT Pin = 1) |

**Table 1-7. Details of Boot mode (For GD32H75E)**

| SCR | SPC[7:0] | BOOT_ADDRESS (configured in BOOT_ADDRx(x = 0,1)) | BOOT_MODE | Boot from |
|---|---|---|---|---|
| 1 | x | XXXX | SECURITY BOOT | ROM |
| 0 | Protection level high | 0x9000_0000 | USER BOOT | OSPI0 |
| | | 0x7000_0000 | USER BOOT | OSPI1 |
| | | 0x0800_0000~max user flash | USER BOOT | BOOT_ADDRESS |
| | | other | USER BOOT | 0x0800_0000 |
| | No protection | 0x9000_0000 | USER BOOT | OSPI0 |
| | | 0x7000_0000 | USER BOOT | OSPI1 |

| SCR | SPC[7:0] | BOOT_ADDRESS (configured in BOOT_ADDRx(x = 0,1)) | BOOT_MODE | Boot from |
|---|---|---|---|---|
| | / Protection level low | 0x2408_0000~ max RAM shared(ITCM/DTCM/AXI) | SRAM BOOT(RAM shared) | BOOT_ADDRESS |
| | | 0x2400_0000~ max AXI SRAM | SRAM BOOT(AXI SRAM) | BOOT_ADDRESS |
| | | 0x2000_0000 | SRAM BOOT(DTCM) | 0x2000_0000 |
| | | 0x0800_0000~max user flash | USER BOOT | BOOT_ADDRESS |
| | | 0x0000_0000 | SRAM BOOT(ITCM) | 0x0000_0000 |
| | | 0x1FF0_0000 | SYSTEM BOOT | BootLoader |
| | | Other | USER BOOT | 0x0800_0000(BOOT Pin = 0) |
| | | | SYSTEM BOOT | BootLoader(BOOT Pin = 1) |

The embedded bootloader supports multi interfaces to update the Flash memory. There will be USART ports, USBHS ports, SDIO ports can be used on GD32H73x_75x line products. The details are shown in the datasheet.

## 1.5. System configuration controller (SYSCFG)

The system configuration controller main functions are the following:

- Analog switch configuration management
- I2C Fm+ configuration
- Selection of the Ethernet PHY interface.
- Management of the external interrupt line connection to the GPIOs
- Management of the I/O compensation cell
- Management BOR reset level
- Management timer break input lock.

## 1.6. Timer break input lock

When internal SRAM/FMC ECC error, LVD or CPU core lockup occurs, this function allows to disable TIMER output. Refer to ***Lockup control register (SYSCFG_LKCTL)*** for details.

## 1.7. AXI interconnect matrix (AXIIM)

The AXI interconnect is based on the Arm® CoreLink™ NIC-400 Network Interconnect. It has 6 initiator ports or ASIBs (AMBA slave interface blocks), and 8 target ports or AMIBs (AMBA master interface blocks).

### 1.7.1. Characteristics

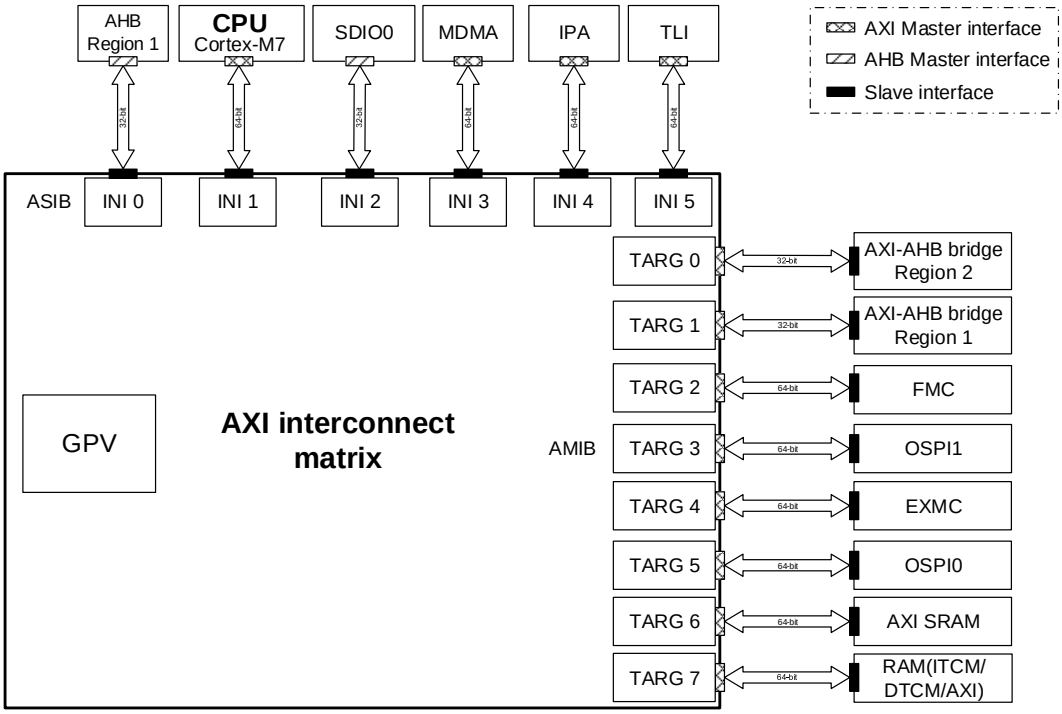
The main functions of AXI interconnect matrix are the following:

- 64-bit AXI bus switch matrix with 6 ASIBs and 8 AMIBs
- Distributed Global Programmers View (GPV)
- Programmable Quality of Service (QoS)

### 1.7.2. Function overview

The block diagram of AXI interconnect matrix is shown in ***Figure 1-9. Block diagram of AXI interconnect matrix***.

**Figure 1-9. Block diagram of AXI interconnect matrix**




The configurations of ASIB and AMIB are shown in ***Table 1-8. Configuration of ASIBs*** and ***Table 1-9. Configuration of AMIBs***.

**Table 1-8. Configuration of ASIBs**

| ASIB | Protocol | Bus width | Read issuing | Write issuing | Master interface |
|---|---|---|---|---|---|
| INI 0 | AHB-lite | 32 | 1 | 1 | AHB Region 1 |
| INI 1 | AXI4 | 64 | 7 | 32 | CPU Cortex-M7 |
| INI 2 | AHB-lite | 32 | 1 | 1 | SDIO0 |
| INI 3 | AXI4 | 64 | 4 | 1 | MDMA |
| INI 4 | AXI4 | 64 | 3 | 1 | IPA |
| INI 5 | AXI4 | 64 | 1 | 1 | TLI |

**Table 1-9. Configuration of AMIBs**

| AMIB | Protocol | Bus width | Read acceptance | Write acceptance | Total acceptance | Slave interface |
|---|---|---|---|---|---|---|
| TARG 0 | AXI4 | 32 | 1 | 1 | 1 | AHB3 Peripherals and Region 2 |
| TARG 1 | AXI4 | 32 | 1 | 1 | 1 | Region 1 |
| TARG 2 | AXI4 | 64 | 3 | 2 | 5 | FMC |
| TARG 3 | AXI4 | 64 | 2 | 1 | 3 | OSPI1 |
| TARG 4 | AXI4 | 64 | 3 | 3 | 6 | EXMC |
| TARG 5 | AXI4 | 64 | 2 | 1 | 3 | OSPI0 |
| TARG 6 | AXI4 | 64 | 2 | 2 | 2 | AXI SRAM |
| TARG 7 | AXI4 | 64 | 2 | 2 | 2 | RAM shared (ITCM/DTCM/AXI SRAM) |

**Quality of Service (QoS)**

Using QoS provides configurable QoS options for ASIB and AMIB, regulation of read and write requests and programmable QoS facilities for attached AMBA masters. The AXI interconnect matrix uses priority based arbitration when different ASIBs try to access a AMIB. The ASIB has configurable read and write channel priority. The priority range is 0x0 ~ 0xF and 0 is the lowest priority. Refer to register ***AXI Slave Port x read QOS control register (AXI_SPx_RDQOS_CTL)*** and ***AXI Slave Port x write QOS control register (AXI_SPx_WRQOS_CTL)*** for details. Least recently used (LRU) priority scheme is used when the priorities of different ASIBs are the same.

**Global Programmers View (GPV)**

Global Programmers View (GPV) for the entire interconnect that is configurable so that any master, or a discrete configuration slave interface, can access it. For more information, see the Arm® CoreLink™ QoS-400 Network Interconnect Advanced Quality of Service, Supplement to Arm® CoreLink™ NIC-400 Network Interconnect Technical Reference Manual.

## 1.8. System configuration registers

SYSCFG base address: 0x5800 0400

### 1.8.1. Peripheral mode configuration register (SYSCFG_PMCFG)

Address offset: 0x004
Reset value: 0x0F00 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | PC3SWON | PC2SWON | PA1SWON | PA0SWON | ENET0_PHY_SEL | ENET1_PHY_SEL | Reserved | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | PB9FMPEN | PB8FMPEN | PB7FMPEN | PB6FMPEN | I2C3FMPEN | I2C2FMPEN | I2C1FMPEN | I2C0FMPEN |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27 | PC3SWON | PC3 switch open<br>This bit controls the analog switch between PC3 and PC3_C, which determines the pads are conneted or separated.<br>0: Close analog switch<br>1: Open analog switch (2 pads are separated pads) |
| 26 | PC2SWON | PC2 switch open<br>This bit controls the analog switch between PC2 and PC2_C, which determines the pads are conneted or separated.<br>0: Close analog switch<br>1: Open analog switch (2 pads are separated pads) |
| 25 | PA1SWON | PA1 switch open<br>This bit controls the analog switch between PA1 and PA1_C, which determines the pads are conneted or separated.<br>0: Close analog switch<br>1: Open analog switch (2 pads are separated pads) |
| 24 | PA0SWON | PA0 switch open<br>This bit controls the analog switch between PA0 and PA0_C, which determines the pads are conneted or separated.<br>0: Close analog switch<br>1: Open analog switch (2 pads are separated pads) |
| 23 | ENET0_PHY_SEL | Ethernet0 PHY interface selection |

| | | |
|---|---|---|
| | | This bit selects the Ethernet PHY interface.<br>0: MII is selected<br>1: RMII is selected |
| 22 | ENET1_PHY_SEL | Ethernet1 PHY interface selection<br>This bit selects the Ethernet PHY interface.<br>0: MII is selected<br>1: RMII is selected |
| 21:8 | Reserved | Must be kept at reset value. |
| 7 | PB9FMPEN | I2C Fm+ mode on PB9 pin enable<br>This bit controls I2C Fm+ mode, the speed control of the pin is bypassed.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 6 | PB8FMPEN | I2C Fm+ mode on PB8 pin enable<br>This bit controls I2C Fm+ mode, the speed control of the pin is bypassed.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 5 | PB7FMPEN | I2C Fm+ mode on PB7 pin enable<br>This bit controls I2C Fm+ mode, the speed control of the pin is bypassed.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 4 | PB6FMPEN | I2C Fm+ mode on PB6 pin enable<br>This bit controls I2C Fm+ mode, the speed control of the pin is bypassed.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 3 | I2C3FMPEN | I2C3 Fm+ mode enable<br>This bit controls I2C3 Fm+ mode.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 2 | I2C2FMPEN | I2C2 Fm+ mode enable<br>This bit controls I2C2 Fm+ mode.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 1 | I2C1FMPEN | I2C1 Fm+ mode enable<br>This bit controls I2C1 Fm+ mode.<br>0: Disable Fm+ mode<br>1: Enable Fm+ mode |
| 0 | I2C0FMPEN | I2C0 Fm+ mode enable<br>This bit controls I2C0 Fm+ mode.<br>0: Disable Fm+ mode |

1: Enable Fm+ mode

### 1.8.2. EXTI sources selection register 0 (SYSCFG_EXTISS0)

Address offset: 0x008

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI3_SS[3:0] | | | | EXTI2_SS[3:0] | | | | EXTI1_SS[3:0] | | | | EXTI0_SS[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:12 | EXTI3_SS[3:0] | EXTI 3 sources selection<br>0000: PA3 pin<br>0001: PB3 pin<br>0010: PC3 pin<br>0011: PD3 pin<br>0100: PE3 pin<br>0101: PF3 pin<br>0110: PG3 pin<br>0111: PH3 pin |
| 11:8 | EXTI2_SS[3:0] | EXTI 2 sources selection<br>0000: PA2 pin<br>0001: PB2 pin<br>0010: PC2 pin<br>0011: PD2 pin<br>0100: PE2 pin<br>0101: PF2 pin<br>0110: PG2 pin<br>0111: PH2 pin<br>1010: PK2 pin |
| 7:4 | EXTI1_SS[3:0] | EXTI 1 sources selection<br>0000: PA1 pin<br>0001: PB1 pin<br>0010: PC1 pin<br>0011: PD1 pin<br>0100: PE1 pin<br>0101: PF1 pin |

| | | |
|---|---|---|
| | | 0110: PG1 pin<br>0111: PH1 pin<br>1010: PK1 pin |
| 3:0 | EXTI0_SS[3:0] | EXTI 0 sources selection<br>0000: PA0 pin<br>0001: PB0 pin<br>0010: PC0 pin<br>0011: PD0 pin<br>0100: PE0 pin<br>0101: PF0 pin<br>0110: PG0 pin<br>0111: PH0 pin<br>1010: PK0 pin |

### 1.8.3. EXTI sources selection register 1 (SYSCFG_EXTISS1)

Ad Address offset: 0x00C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI7_SS[3:0] | | | | EXTI6_SS[3:0] | | | | EXTI5_SS[3:0] | | | | EXTI4_SS[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:12 | EXTI7_SS[3:0] | EXTI 7 sources selection<br>0000: PA7 pin<br>0001: PB7 pin<br>0010: PC7 pin<br>0011: PD7 pin<br>0100: PE7 pin<br>0101: PF7 pin<br>0110: PG7 pin<br>0111: PH7 pin |
| 11:8 | EXTI6_SS[3:0] | EXTI 6 sources selection<br>0000: PA6 pin<br>0001: PB6 pin<br>0010: PC6 pin<br>0011: PD6 pin |

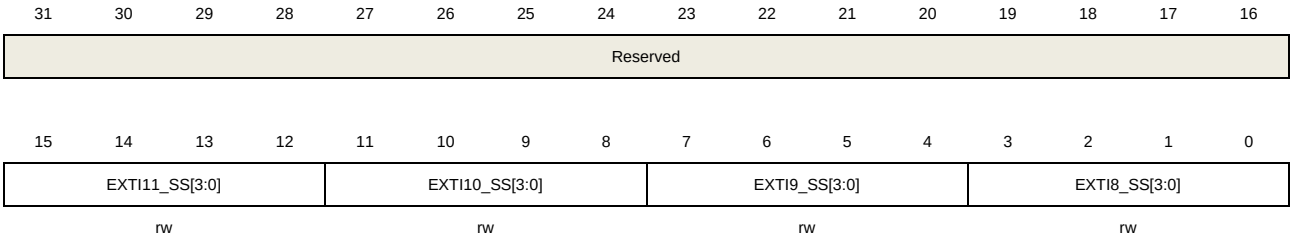
| | | |
|---|---|---|
| | | 0100: PE6 pin<br>0101: PF6 pin<br>0110: PG6 pin<br>0111: PH6 pin |
| 7:4 | EXTI5_SS[3:0] | EXTI 5 sources selection<br>0000: PA5 pin<br>0001: PB5 pin<br>0010: PC5 pin<br>0011: PD5 pin<br>0100: PE5 pin<br>0101: PF5 pin<br>0110: PG5 pin<br>0111: PH5 pin |
| 3:0 | EXTI4_SS[3:0] | EXTI 4 sources selection<br>0000: PA4 pin<br>0001: PB4 pin<br>0010: PC4 pin<br>0011: PD4 pin<br>0100: PE4 pin<br>0101: PF4 pin<br>0110: PG4 pin<br>0111: PH4 pin |

### 1.8.4. EXTI sources selection register 2 (SYSCFG_EXTISS2)

Address offset: 0x010

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:12 | EXTI11_SS[3:0] | EXTI 11 sources selection<br>0000: Reserved<br>0001: PB11 pin<br>0010: PC11 pin<br>0011: PD11 pin |

| | | |
|---|---|---|
| | | 0100: PE11 pin<br>0101: PF11 pin<br>0110: PG11 pin<br>0111: PH11 pin<br>1001: PJ11 pin |
| 11:8 | EXTI10_SS[3:0] | EXTI 10 sources selection<br>0000: PA10 pin<br>0001: PB10 pin<br>0010: PC10 pin<br>0011: PD10 pin<br>0100: PE10 pin<br>0101: PF10 pin<br>0110: PG10 pin<br>0111: PH10 pin<br>1001: PJ10 pin |
| 7:4 | EXTI9_SS[3:0] | EXTI 9 sources selection<br>0000: PA9 pin<br>0001: PB9 pin<br>0010: PC9 pin<br>0011: PD9 pin<br>0100: PE9 pin<br>0101: PF9 pin<br>0110: PG9 pin<br>0111: PH9 pin<br>1001: PJ9 pin |
| 3:0 | EXTI8_SS[3:0] | EXTI 8 sources selection<br>0000: PA8 pin<br>0001: PB8 pin<br>0010: PC8 pin<br>0011: PD8 pin<br>0100: PE8 pin<br>0101: PF8 pin<br>0110: PG8 pin<br>0111: PH8 pin<br>1001: PJ8 pin |

### 1.8.5. EXTI sources selection register 3 (SYSCFG_EXTISS3)

Address offset: 0x014

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTI15_SS[3:0] | | | | EXTI14_SS[3:0] | | | | EXTI13_SS[3:0] | | | | EXTI12_SS[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:12 | EXTI15_SS[3:0] | EXTI 15 sources selection<br>0000: PA15 pin<br>0001: PB15 pin<br>0010: PC15 pin<br>0011: PD15 pin<br>0100: PE15 pin<br>0101: PF15 pin<br>0110: PG15 pin<br>0111: PH15 pin |
| 11:8 | EXTI14_SS[3:0] | EXTI 14 sources selection<br>0000: PA14 pin<br>0001: PB14 pin<br>0010: PC14 pin<br>0011: PD14 pin<br>0100: PE14 pin<br>0101: PF14 pin<br>0110: PG14 pin<br>0111: PH14 pin |
| 7:4 | EXTI13_SS[3:0] | EXTI 13 sources selection<br>0000: PA13 pin<br>0001: PB13 pin<br>0010: PC13 pin<br>0011: PD13 pin<br>0100: PE13 pin<br>0101: PF13 pin<br>0110: PG13 pin<br>0111: PH13 pin |
| 3:0 | EXTI12_SS[3:0] | EXTI 12 sources selection<br>0000: Reserved<br>0001: PB12 pin<br>0010: PC12 pin<br>0011: PD12 pin<br>0100: PE12 pin<br>0101: PF12 pin |

0110: PG12 pin

0111: PH12 pin

### 1.8.6. Lockup control register (SYSCFG_LKCTL)

Address offset: 0x018

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AXIRAM_LOCK | ITCM_LOCK | DTCM_LOCK | SRAM0_LOCK | SRAM1_LOCK | Reserved | | | BKPRAM_LOCK | CPU_LOCK | Reserved | | | LVD_LOCK | Reserved | |
| rw | rw | rw | rw | rw | | | | rw | rw | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | AXIRAM_LOCK | Region 0 AXI-SRAM ECC double error lock bit<br>This bit is set by software and cleared by a system reset.<br>0: Region 0 AXI-SRAM ECC double error signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: Region 0 AXI-SRAM ECC double error signal is connected from the break input of TIMER0/7/14/15/16 |
| 14 | ITCM_LOCK | Region 0 ITCM-RAM ECC double error lock bit<br>This bit is set by software and cleared by a system reset.<br>0: Region 0 ITCM-RAM ECC double error signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: Region 0 ITCM-RAM ECC double error signal is connected from the break input of TIMER0/7/14/15/16 |
| 13 | DTCM_LOCK | Region 0 DTCM ECC double error lock bit<br>This bit is set by software and cleared by a system reset.<br>0: Region 0 DTCM ECC double error signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: Region 0 DTCM ECC double error signal is connected from the break input of TIMER0/7/14/15/16 |
| 12 | SRAM0_LOCK | Region 1 SRAM0 ECC double error lockup bit<br>This bit is set by software and cleared by a system reset.<br>0: Region 1 SRAM0 ECC double error signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: Region 1 SRAM0 ECC double error signal is connected from the break input of |

| | | |
|---|---|---|
| | | TIMER0/7/14/15/16 |
| 11 | SRAM1_LOCK | Region 1 SRAM1 ECC double error lockup bit<br>This bit is set by software and cleared by a system reset.<br>0: Region 1 SRAM1 ECC double error signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: Region 1 SRAM1 ECC double error signal is connected from the break input of TIMER0/7/14/15/16 |
| 10:8 | Reserved | Must be kept at reset value. |
| 7 | BKPRAM_LOCK | Region 2 backup SRAM ECC double error lockup bit<br>This bit is set by software and cleared by a system reset.<br>0: Region 2 backup SRAM ECC double error signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: Region 2 backup SRAM ECC double error signal is connected from the break input of TIMER0/7/14/15/16 |
| 6 | CPU_LOCK | CPU lockup bit<br>This bit is set by software and cleared by a system reset.<br>0: CPU lockup signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: CPU lockup signal is connected from the break input of TIMER0/7/14/15/16 |
| 5:3 | Reserved | Must be kept at reset value. |
| 2 | LVD_LOCK | Low voltage detector lockup bit<br>This bit is set by software and cleared by a system reset.<br>0: LVD signal is disconnected from the break input of TIMER0/7/14/15/16<br>1: LVD signal is connected from the break input of TIMER0/7/14/15/16 |
| 1:0 | Reserved | Must be kept at reset value. |

### 1.8.7. I/O compensation control register (SYSCFG_CPSCTL)

Address offset: 0x020

Reset value: 0x00X0 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | IOLV | Reserved | | | | | | IOSPDOP |
| | | | | | | | | r | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CPS_RDY | Reserved | | | | | | | CPS_EN |
| | | | | | | | r | | | | | | | | rw |

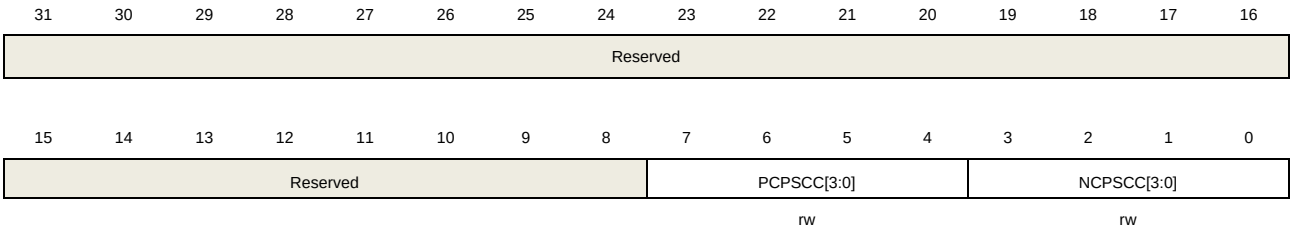
| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 23 | IOLV | I/O in low voltage state<br>0: Product supply voltage is working higher than 2.5V<br>1: Product supply voltage is working below 2.5V |
| 22:17 | Reserved | Must be kept at reset value. |
| 16 | IOSPDOP | I/O speed optimization, High-speed at low-voltage<br>This bit is written by software to optimize the I/O speed when the product voltage is low.<br>It must be used only if the product supply voltage is below 2.5V (IOLV bit is '1').<br>Setting this bit when $V_{DD}$ is higher than 2.5V might be destructive.<br>0: No I/O speed optimization<br>1: I/O speed optimization |
| 15:9 | Reserved | Must be kept at reset value. |
| 8 | CPS_RDY | Compensation cell ready flag<br>This bit provides the status of the compensation cell.<br>0: I/O compensation cell not ready<br>1: I/O compensation cell ready |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | CPS_EN | I/O compensation cell enable<br>This bit enables the I/O compensation cell.<br>0: I/O compensation cell disabled<br>1: I/O compensation cell enabled |

### 1.8.8. I/O compensation cell code configuration register (SYSCFG_CPSCCCFG)

Address offset: 0x028
Reset value: 0x0000 0088

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | PCPSCC[3:0] | PMOS compensation cell code<br>These bits define I/O compensation cell code for PMOS transistors. |

| | | |
|---|---|---|
| 3:0 | NCPSCC[3:0] | NMOS compensation cell code<br>These bits define I/O compensation cell code for NMOS transistors. |

### 1.8.9. Timer input selection register 0 (SYSCFG_TIMERCISEL0)

Address offset: 0x034

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMER0_CI3_SEL[3:0] | | | | TIMER0_CI2_SEL[3:0] | | | | TIMER0_CI1_SEL[3:0] | | | | TIMER0_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER7_CI3_SEL[3:0] | | | | TIMER7_CI2_SEL[3:0] | | | | TIMER7_CI1_SEL[3:0] | | | | TIMER7_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | TIMER0_CI3_SEL[3:0] | Selects TIMER0_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER0_CH3 input<br>Others: Reserved |
| 27:24 | TIMER0_CI2_SEL[3:0] | Selects TIMER0_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER0_CH2 input<br>Others: Reserved |
| 23:20 | TIMER0_CI1_SEL[3:0] | Selects TIMER0_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER0_CH1 input<br>Others: Reserved |
| 19:16 | TIMER0_CI0_SEL[3:0] | Selects TIMER0_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER0_CH0 input<br>0001: CMP0 output<br>Others: Reserved |
| 15:12 | TIMER7_CI3_SEL[3:0] | Selects TIMER7_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER7_CH3 input<br>Others: Reserved |
| 11:8 | TIMER7_CI2_SEL[3:0] | Selects TIMER7_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER7_CH2 input |

| | | |
|---|---|---|
| | | Others: Reserved |
| 7:4 | TIMER7_CI1_SEL[3:0] | Selects TIMER7_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER7_CH1 input<br>Others: Reserved |
| 3:0 | TIMER7_CI0_SEL[3:0] | Selects TIMER7_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER7_CH0 input<br>0001: CMP1 output<br>Others: Reserved |

## 1.8.10. Timer input selection register 1 (SYSCFG_TIMERCISEL1)

Address offset: 0x038

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMER1_CI3_SEL[3:0] | | | | TIMER1_CI2_SEL[3:0] | | | | TIMER1_CI1_SEL[3:0] | | | | TIMER1_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER2_CI3_SEL[3:0] | | | | TIMER2_CI2_SEL[3:0] | | | | TIMER2_CI1_SEL[3:0] | | | | TIMER2_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | TIMER1_CI3_SEL[3:0] | TIMER1_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER1_CH3 input<br>0001: CMP0 output<br>0010: CMP1 output<br>0011: CMP0 output or CMP1 output<br>Others: Reserved |
| 27:24 | TIMER1_CI2_SEL[3:0] | TIMER1_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER1_CH2 input<br>Others: Reserved |
| 23:20 | TIMER1_CI1_SEL[3:0] | TIMER1_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER1_CH1 input<br>Others: Reserved |
| 19:16 | TIMER1_CI0_SEL[3:0] | TIMER1_CI0 input selection<br>These bits select the TIMER input source. |

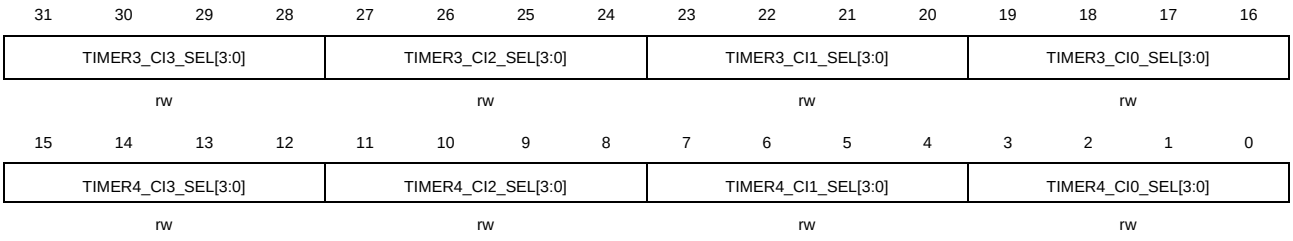
| | | |
|---|---|---|
| | | 0000: TIMER1_CH0 input<br>Others: Reserved |
| 15:12 | TIMER2_CI3_SEL[3:0] | TIMER2_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER2_CH3 input<br>Others: Reserved |
| 11:8 | TIMER2_CI2_SEL[3:0] | TIMER2_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER2_CH2 input<br>Others: Reserved |
| 7:4 | TIMER2_CI1_SEL[3:0] | TIMER2_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER2_CH1 input<br>Others: Reserved |
| 3:0 | TIMER2_CI0_SEL[3:0] | TIMER2_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER2_CH0 input<br>0001: CMP0 output<br>0010: CMP1 output<br>0011: CMP0 output or CMP1 output<br>Others: Reserved |

### 1.8.11. Timer input selection register 2 (SYSCFG_TIMERCISEL2)

Address offset: 0x03C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | TIMER3_CI3_SEL[3:0] | TIMER3_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER3_CH3 input<br>Others: Reserved |
| 27:24 | TIMER3_CI2_SEL[3:0] | TIMER3_CI2 input selection<br>These bits select the TIMER input source. |

0000: TIMER3_CH2 input
Others: Reserved

| | | |
|---|---|---|
| 23:20 | TIMER3_CI1_SEL[3:0] | TIMER3_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER3_CH1 input<br>Others: Reserved |
| 19:16 | TIMER3_CI0_SEL[3:0] | TIMER3_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER3_CH0 input<br>Others: Reserved |
| 15:12 | TIMER4_CI3_SEL[3:0] | TIMER4_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER4_CH3 input<br>Others: Reserved |
| 11:8 | TIMER4_CI2_SEL[3:0] | TIMER4_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER4_CH2 input<br>Others: Reserved |
| 7:4 | TIMER4_CI1_SEL[3:0] | TIMER4_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER4_CH1 input<br>Others: Reserved |
| 3:0 | TIMER4_CI0_SEL[3:0] | TIMER4_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER4_CH0 input<br>Others: Reserved |

## 1.8.12. Timer input selection register 3 (SYSCFG_TIMERCISEL3)

Address offset: 0x040
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMER22_CI3_SEL[3:0] | | | | TIMER22_CI2_SEL[3:0] | | | | TIMER22_CI1_SEL[3:0] | | | | TIMER22_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER23_CI3_SEL[3:0] | | | | TIMER23_CI2_SEL[3:0] | | | | TIMER23_CI1_SEL[3:0] | | | | TIMER23_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:28 | TIMER22_CI3_SEL[3:0] | TIMER22_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER22_CH3 input<br>0001: CMP0 output<br>0010: CMP1 output<br>0011: CMP0 output or CMP1 output<br>Others: Reserved |
| 27:24 | TIMER22_CI2_SEL[3:0] | TIMER22_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER22_CH2 input<br>Others: Reserved |
| 23:20 | TIMER22_CI1_SEL[3:0] | TIMER22_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER22_CH1 input<br>Others: Reserved |
| 19:16 | TIMER22_CI0_SEL[3:0] | TIMER22_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER22_CH0 input<br>Others: Reserved |
| 15:12 | TIMER23_CI3_SEL[3:0] | TIMER23_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER23_CH3 input<br>Others: Reserved |
| 11:8 | TIMER23_CI2_SEL[3:0] | TIMER23_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER23_CH2 input<br>Others: Reserved |
| 7:4 | TIMER23_CI1_SEL[3:0] | TIMER23_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER23_CH1 input<br>Others: Reserved |
| 3:0 | TIMER23_CI0_SEL[3:0] | TIMER23_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER23_CH0 input<br>Others: Reserved |

### 1.8.13. Timer input selection register 4 (SYSCFG_TIMERCISEL4)

Address offset: 0x044
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMER30_CI3_SEL[3:0] | | | | TIMER30_CI2_SEL[3:0] | | | | TIMER30_CI1_SEL[3:0] | | | | TIMER30_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER31_CI3_SEL[3:0] | | | | TIMER31_CI2_SEL[3:0] | | | | TIMER31_CI1_SEL[3:0] | | | | TIMER31_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | TIMER30_CI3_SEL[3:0] | TIMER30_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER30_CH3 input<br>Others: Reserved |
| 27:24 | TIMER30_CI2_SEL[3:0] | TIMER30_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER30_CH2 input<br>Others: Reserved |
| 23:20 | TIMER30_CI1_SEL[3:0] | TIMER30_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER30_CH1 input<br>Others: Reserved |
| 19:16 | TIMER30_CI0_SEL[3:0] | TIMER30_CI0 input selection<br>These bits select the TIMER input source.<br>0000: TIMER30_CH0 input<br>0001: CMP0 output<br>0010: CMP1 output<br>0011: CMP0 output or CMP1 output<br>Others: Reserved |
| 15:12 | TIMER31_CI3_SEL[3:0] | TIMER31_CI3 input selection<br>These bits select the TIMER input source.<br>0000: TIMER31_CH3 input<br>Others: Reserved |
| 11:8 | TIMER31_CI2_SEL[3:0] | TIMER31_CI2 input selection<br>These bits select the TIMER input source.<br>0000: TIMER31_CH2 input<br>Others: Reserved |
| 7:4 | TIMER31_CI1_SEL[3:0] | TIMER31_CI1 input selection<br>These bits select the TIMER input source.<br>0000: TIMER31_CH1 input<br>Others: Reserved |
| 3:0 | TIMER31_CI0_SEL[3:0] | TIMER31_CI0 input selection<br>These bits select the TIMER input source. |

0000: TIMER31_CH0 input
0001: CMP0 output
0010: CMP1 output
0011: CMP0 output or CMP1 output
Others: Reserved

### 1.8.14. Timer input selection register 5 (SYSCFG_TIMERCISEL5)

Address offset: 0x048
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMER42_CI1_SEL[3:0] | | | | TIMER42_CI0_SEL[3:0] | | | | TIMER41_CI1_SEL[3:0] | | | | TIMER41_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TIMER40_CI1_SEL[3:0] | | | | TIMER40_CI0_SEL[3:0] | | | | TIMER14_CI1_SEL[3:0] | | | | TIMER14_CI0_SEL[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | TIMER42_CI1_SEL[3:0] | Selects TIMER42_CI1 input<br>These bits select the TIMER input source.<br>0000: TIMER42_CH1 input<br>0001: TIMER4_CH1 input<br>0010: TIMER22_CH1 input<br>0011: TIMER23_CH1 input<br>Others: Reserved |
| 27:24 | TIMER42_CI0_SEL[3:0] | Selects TIMER42_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER42_CH0 input<br>0001: TIMER4_CH0 input<br>0010: TIMER22_CH0 input<br>0011: TIMER23_CH0 input<br>0100: LXTAL<br>0101: LPIRC4M<br>0110: CKOUT1<br>Others: Reserved |
| 23:20 | TIMER41_CI1_SEL[3:0] | Selects TIMER41_CI1 input<br>These bits select the TIMER input source.<br>0000: TIMER41_CH1 input<br>0001: TIMER3_CH1 input<br>0010: TIMER4_CH1 input<br>0011: TIMER22_CH1 input |

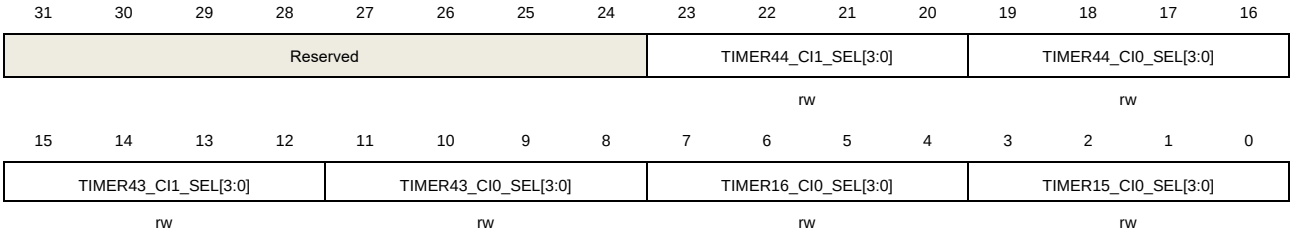
| | | |
|---|---|---|
| | | Others: Reserved |
| 19:16 | TIMER41_CI0_SEL[3:0] | Selects TIMER41_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER41_CH0 input<br>0001: TIMER3_CH0 input<br>0010: TIMER4_CH0 input<br>0011: TIMER22_CH0 input<br>0100: LXTAL<br>0101: LPIRC4M<br>0110: CKOUT1<br>Others: Reserved |
| 15:12 | TIMER40_CI1_SEL[3:0] | Selects TIMER40_CI1 input<br>These bits select the TIMER input source.<br>0000: TIMER40_CH1 input<br>0001: TIMER2_CH1 input<br>0010: TIMER3_CH1 input<br>0011: TIMER4_CH1 input<br>Others: Reserved |
| 11:8 | TIMER40_CI0_SEL[3:0] | Selects TIMER40_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER40_CH0 input<br>0001: TIMER2_CH0 input<br>0010: TIMER3_CH0 input<br>0011: TIMER4_CH0 input<br>0100: LXTAL<br>0101: LPIRC4M<br>0110: CKOUT1<br>Others: Reserved |
| 7:4 | TIMER14_CI1_SEL[3:0] | Selects TIMER14_CI1 input<br>These bits select the TIMER input source.<br>0000: TIMER14_CH1 input<br>0001: TIMER1_CH1 input<br>0010: TIMER2_CH1 input<br>0011: TIMER3_CH1 input<br>Others: Reserved |
| 3:0 | TIMER14_CI0_SEL[3:0] | Selects TIMER14_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER14_CH0 input<br>0001: TIMER1_CH0 input<br>0010: TIMER2_CH0 input<br>0011: TIMER3_CH0 input<br>0100: LXTAL |

0101: LPIRC4M
0110: CKOUT1
Others: Reserved

### 1.8.15. Timer input selection register 6 (SYSCFG_TIMERCISEL6)

Address offset: 0x04C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:20 | TIMER44_CI1_SEL[3:0] | Selects TIMER44_CI1 input<br>These bits select the TIMER input source.<br>0000: TIMER44_CH1 input<br>0001: TIMER23_CH1 input<br>0010: TIMER30_CH1 input<br>0011: TIMER31_CH1 input<br>Others: Reserved |
| 19:16 | TIMER44_CI0_SEL[3:0] | Selects TIMER44_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER44_CH0 input<br>0001: TIMER23_CH0 input<br>0010: TIMER30_CH0 input<br>0011: TIMER31_CH0 input<br>0100: LXTAL<br>0101: LPIRC4M<br>0110: CKOUT1<br>Others: Reserved |
| 15:12 | TIMER43_CI1_SEL[3:0] | Selects TIMER43_CI1 input<br>These bits select the TIMER input source.<br>0000: TIMER43_CH1 input<br>0001: TIMER22_CH1 input<br>0010: TIMER23_CH1 input<br>0011: TIMER30_CH1 input |

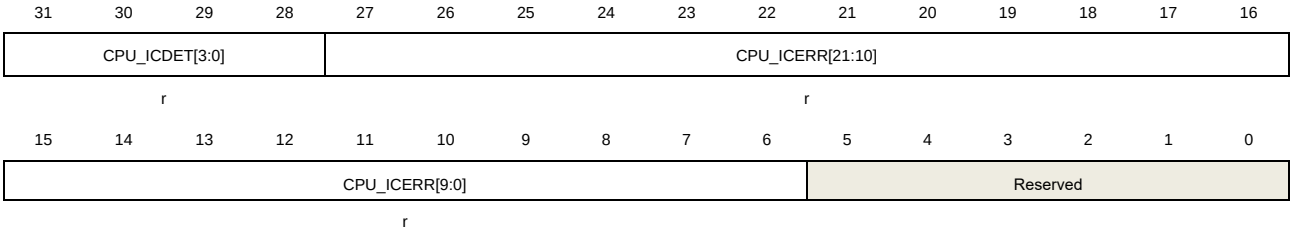
| | | |
|---|---|---|
| | | Others: Reserved |
| 11:8 | TIMER43_CI0_SEL[3:0] | Selects TIMER43_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER43_CH0 input<br>0001: TIMER22_CH0 input<br>0010: TIMER23_CH0 input<br>0011: TIMER30_CH0 input<br>0100: LXTAL<br>0101: LPIRC4M<br>0110: CKOUT1<br>Others: Reserved |
| 7:4 | TIMER16_CI0_SEL[3:0] | Selects TIMER16_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER16_CH0 input<br>0001: RSPDIF symbol_ck<br>0010: CK_HXTAL / RTCDIV<br>0011: CKOUT0<br>Others: Reserved |
| 3:0 | TIMER15_CI0_SEL[3:0] | Selects TIMER15_CI0 input<br>These bits select the TIMER input source.<br>0000: TIMER15_CH0 input<br>0001: IRC32K<br>0010: LXTAL<br>0011: WKUP_IT<br>Others: Reserved |

## 1.8.16. CPU ICACHE error status register(SYSCFG_CPUICAC)

Address offset: 0x054

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



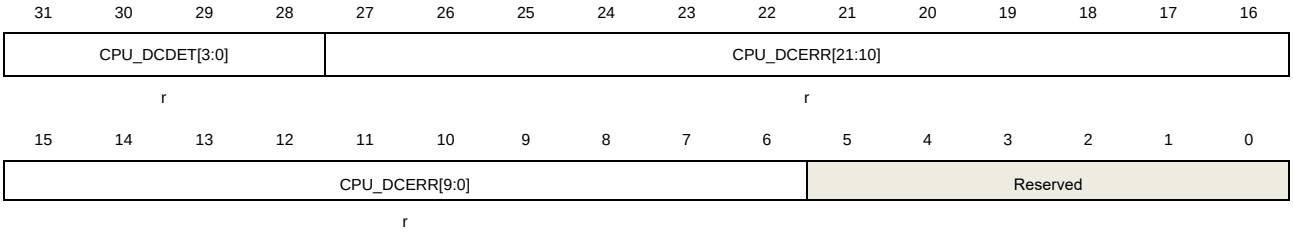

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | CPU_ICDET[3:0] | The ICACHE error detection information<br>These bits are provided by the CPU to indicate the ICACHE error detection |

| | | |
|---|---|---|
| | | information. |
| 27:6 | CPU_ICERR[21:0] | The ICACHE error bank information<br>These bits are provided by the CPU to indicate the ICACHE error bank information. |
| 5:0 | Reserved | Must be kept at reset value. |

## 1.8.17. CPU DCACHE error status register (SYSCFG_CPUDCAC)

Address offset: 0x058
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

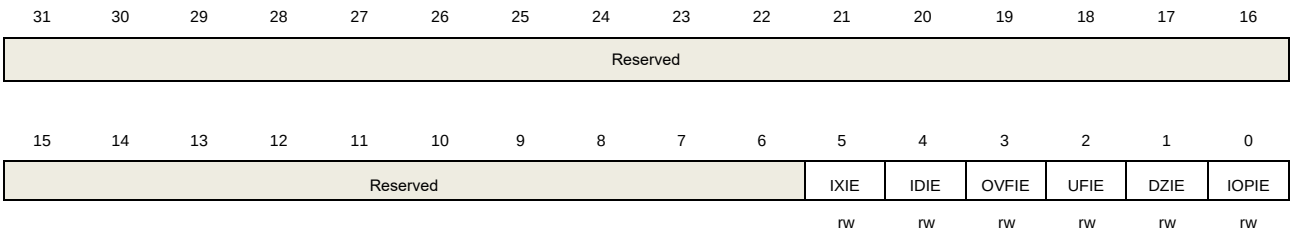



| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | CPU_DCDET[3:0] | The DCACHE error detection information<br>These bits are provided by the CPU to indicate the DCACHE error detection information. |
| 27:6 | CPU_DCERR[21:0] | The DCACHE error bank information<br>These bits are provided by the CPU to indicate the DCACHE error bank information. |
| 5:0 | Reserved | Must be kept at reset value. |

## 1.8.18. FPU interrupt enable register (SYSCFG_FPUINTEN)

Address offset: 0x05C
Reset value: 0x0000 001F

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5 | IXIE | Inexact interrupt enable bit |

| | | |
|---|---|---|
| | | 0: Inexact interrupt disable<br>1: Inexact interrupt enable |
| 4 | IDIE | Input denormal interrupt enable bit<br>0: Input denormal interrupt disable<br>1: Input denormal interrupt enable |
| 3 | OVFIE | Overflow interrupt enable bit<br>0: Overflow interrupt disable<br>1: Overflow interrupt enable |
| 2 | UFIE | Underflow interrupt enable bit<br>0: Underflow interrupt disable<br>1: Underflow interrupt enable |
| 1 | DZIE | Divide by 0 interrupt enable bit<br>0: Divide by 0 interrupt disable<br>1: Divide by 0 interrupt enable |
| 0 | IOPIE | Invalid operation interrupt enable bit<br>0: Invalid operation interrupt disable<br>1: Invalid operation interrupt enable |

### 1.8.19. SRAM configuration register 0 (SYSCFG_SRAMCFG0)

Address offset: 0x64

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | SECURE_SRAM_SIZE[1:0] | |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1:0 | SECURE_SRAM_SIZE[1:0] | Size of secure SRAM<br>These bits are set and cleared by software.<br>00: 0 Kbytes<br>01: 32 Kbytes<br>10: 64 Kbytes<br>11: 128 Kbytes |

### 1.8.20. SRAM configuration register 1 (SYSCFG_SRAMCFG1)

Address offset: 0x68

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | TCM_WAITSTATE |
| | | | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | TCM_WAITSTATE | TCM wait state configuration<br>This bit is set and cleared by software. It can be reset by system reset only.<br>This bit is used to insert wait-state in ITCM / D0TCM / D1TCM.<br>**Note:** When the system clock frequency is higher than $f_{TWW}$, this bit must be set.<br>0: No wait-state<br>1: Insert wait-state |

### 1.8.21. TIMERx configuration register 0 (SYSCFG_TIMERxCFG0, x=0, 7)

Address offset: 0x100 for TIMER0

Address offset: 0x13C for TIMER7

Reset value: 0x0000 0000

TSCFG0[4:0], TSCFG1[4:0]..TSCFG9[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TSCFG5[4:0] | | | | | TSCFG4[4:0] | | | | | TSCFG3[4:0] | | | | |
| | rw | | | | | rw | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TSCFG2[4:0] | | | | | TSCFG1[4:0] | | | | | TSCFG0[4:0] | | | | |
| | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:26 | TSCFG5[4:0] | Event mode configuration<br>A rising edge of the trigger input enables the counter. |

00000: Event mode disable
00001: Internal trigger input 0 (ITI0)
00010: Internal trigger input 1 (ITI1)
00011: Internal trigger input 2 (ITI2)
00100: Internal trigger input 3 (ITI3)
00101: CI0 edge flag (CI0F_ED)
00110: The filtered output of channel 0 input (CI0FE0)
00111: The filtered output of channel 1 input (CI1FE1)
01000: The filtered output of external trigger input (ETIFP)
01001: The filtered output of channel 2 input (CI2FE2)
01010: The filtered output of channel 3 input (CI3FE3)
01011: The filtered output of multi mode channel 0 input (MCI0FEM0)
01100: The filtered output of multi mode channel 1 input (MCI1FEM1)
01101: The filtered output of multi mode channel 2 input (MCI2FEM2)
01110: The filtered output of multi mode channel 3 input (MCI3FEM3)
01111: Reserved
10000: Reserved
10001: Internal trigger input 12 (ITI12)
10010: Internal trigger input 13 (ITI13)
10011: Internal trigger input 14 (ITI14)
Others: Reserved

25:21 TSCFG4[4:0] Pause mode configuration

The trigger input enables the counter clock when it is high and disables the counter when it is low when these bits are not 0.
00000: Pause mode disable
00001: Internal trigger input 0 (ITI0)
00010: Internal trigger input 1 (ITI1)
00011: Internal trigger input 2 (ITI2)
00100: Internal trigger input 3 (ITI3)
00101: Reserved
00110: The filtered output of channel 0 input (CI0FE0)
00111: The filtered output of channel 1 input (CI1FE1)
01000: The filtered output of external trigger input (ETIFP)
01001: The filtered output of channel 2 input (CI2FE2)
01010: The filtered output of channel 3 input (CI3FE3)
01011: The filtered output of multi mode channel 0 input (MCI0FEM0)
01100: The filtered output of multi mode channel 1 input (MCI1FEM1)
01101: The filtered output of multi mode channel 2 input (MCI2FEM2)
01110: The filtered output of multi mode channel 3 input (MCI3FEM3)
01111: Reserved
10000: Reserved
10001: Internal trigger input 12 (ITI12)
10010: Internal trigger input 13 (ITI13)
10011: Internal trigger input 14 (ITI14)

| | | |
|---|---|---|
| | | Others: Reserved |
| 20:16 | TSCFG3[4:0] | Restart mode configuration<br>The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input when these bits are not 0.<br>00000: Restart mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: The filtered output of channel 2 input (CI2FE2)<br>01010: The filtered output of channel 3 input (CI3FE3)<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>01100: The filtered output of multi mode channel 1 input (MCI1FEM1)<br>01101: The filtered output of multi mode channel 2 input (MCI2FEM2)<br>01110: The filtered output of multi mode channel 3 input (MCI3FEM3)<br>01111: Reserved<br>10000: Reserved<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 15 | Reserved | Must be kept at reset value. |
| 14:10 | TSCFG2[4:0] | Quadrature decoder mode 2 configuration<br>00000: Quadrature decoder mode 2 disable<br>Others: The counter counts on both CI0FE0 and CI1FE1 edges, while the direction depends on each other |
| 9:5 | TSCFG1[4:0] | Quadrature decoder mode 1 configuration<br>00000: Quadrature decoder mode 1 disable<br>Others: The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level |
| 4:0 | TSCFG0[4:0] | Quadrature decoder mode 0 configuration<br>00000: Quadrature decoder mode 0 disable<br>Others: The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level. |

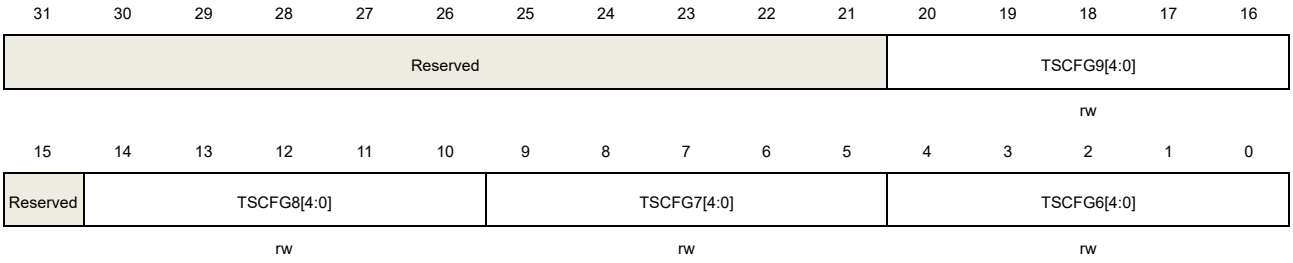
### 1.8.22. TIMERx configuration register 1 (SYSCFG_TIMERxCFG1, x=0, 7)

Address offset: 0x104 for TIMER0

Address offset: 0x140 for TIMER7
Reset value: 0x0000 0000

TSCFG0[4:0], TSCFG1[4:0]..TSCFG9[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | TSCFG9[4:0] | Non-quadrature decoder mode 1 configuration<br>00000: Non-quadrature decoder mode 1 disable<br>Others: The CI0 signal is used as the count pulse( with the CH0P is used to select the counter edge) and the CI1 signal is used as the count direction selection. |
| 15 | Reserved | Must be kept at reset value. |
| 14:10 | TSCFG8[4:0] | Non-quadrature decoder mode 0 configuration<br>00000: Non-quadrature decoder mode 0 disable<br>Others: The CI0 signal is used as the count pulse and CI1 is used as the count selection signal. When CH1P=0, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is high; When CH1P=1, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is low. |
| 9:5 | TSCFG7[4:0] | Restart + event mode configuration<br>The counter is reinitialized and started, the shadow registers are updated on the rising edge of the selected trigger input when these bits are not 0.<br>00000: Restart + event mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: The filtered output of channel 2 input (CI2FE2)<br>01010: The filtered output of channel 3 input (CI3FE3)<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0) |

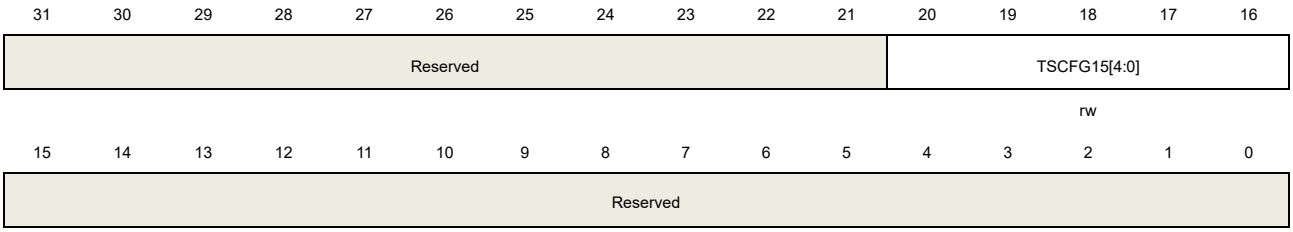
| | | |
|---|---|---|
| | | 01100: The filtered output of multi mode channel 1 input (MCI1FEM1)<br>01101: The filtered output of multi mode channel 2 input (MCI2FEM2)<br>01110: The filtered output of multi mode channel 3 input (MCI3FEM3)<br>01111: Reserved<br>10000: Reserved<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 4:0 | TSCFG6[4:0] | External clock mode 0 configuration<br>The counter counts on the rising edges of the selected trigger when these bits are not 0.<br>00000: External clock mode 0 disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: The filtered output of channel 2 input (CI2FE2)<br>01010: The filtered output of channel 3 input (CI3FE3)<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>01100: The filtered output of multi mode channel 1 input (MCI1FEM1)<br>01101: The filtered output of multi mode channel 2 input (MCI2FEM2)<br>01110: The filtered output of multi mode channel 3 input (MCI3FEM3)<br>01111: Reserved<br>10000: Reserved<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |

### 1.8.23. TIMERx configuration register 2 (SYSCFG_TIMERxCFG2, x=0, 7)

Address offset: 0x108 for TIMER0
Address offset: 0x144 for TIMER7
Reset value: 0x0000 0000

TSCFG0[4:0], TSCFG1[4:0]..TSCFG9[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | TSCFG15[4:0] | Internal trigger input source configuration<br>00000: Internal trigger input 0 (ITI0)<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: Reserved<br>00111: Reserved<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: Reserved<br>01100: Reserved<br>01101: Reserved<br>01110: Reserved<br>01111: Reserved<br>10000: Reserved<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved<br>**Note:** When TSCFG15[4:0] is used, TSCFGy[4:0](y=0..9) should be zero, otherwise, the ITS trigger cooperate with TSCFGy[4:0], and input source depend on TSCFGy[4:0]. |
| 15:0 | Reserved | Must be kept at reset value. |

### 1.8.24. TIMERx configuration register 0 (SYSCFG_TIMERxCFG0, x=1, 2, 3, 4, 22, 23, 30, 31)

Address offset: 0x10C for TIMER1
Address offset: 0x118 for TIMER2
Address offset: 0x124 for TIMER3

Address offset: 0x130 for TIMER4
Address offset: 0x154 for TIMER22
Address offset: 0x160 for TIMER23
Address offset: 0x16C for TIMER30
Address offset: 0x178 for TIMER31
Reset value: 0x0000 0000

TSCFG0[4:0], TSCFG1[4:0]..TSCFG9[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TSCFG5[4:0] | | | | | TSCFG4[4:0] | | | | | TSCFG3[4:0] | | | | |
| | rw | | | | | rw | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TSCFG2[4:0] | | | | | TSCFG1[4:0] | | | | | TSCFG0[4:0] | | | | |
| | rw | | | | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:26 | TSCFG5[4:0] | Event mode configuration<br>A rising edge of the trigger input enables the counter.<br>00000: Event mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: Internal trigger input 4 (ITI4)<br>01010: Internal trigger input 5 (ITI5)<br>01011: Reserved<br>01100: Internal trigger input 7 (ITI7)<br>01101: Reserved<br>01110: Internal trigger input 9 (ITI9)<br>01111: Internal trigger input 10 (ITI10)<br>10000: Internal trigger input 11 (ITI11)<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 25:21 | TSCFG4[4:0] | Pause mode configuration |

| | | |
|---|---|---|
| | | The trigger input enables the counter clock when it is high and disables the counter when it is low when these bits are not 0.<br>00000: Pause mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: Reserved<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: Internal trigger input 4 (ITI4)<br>01010: Internal trigger input 5 (ITI5)<br>01011: Reserved<br>01100: Internal trigger input 7 (ITI7)<br>01101: Reserved<br>01110: Internal trigger input 9 (ITI9)<br>01111: Internal trigger input 10 (ITI10)<br>10000: Internal trigger input 11 (ITI11)<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 20:16 | TSCFG3[4:0] | Restart mode configuration<br>The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input when these bits are not 0.<br>00000: Restart mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: Internal trigger input 4 (ITI4)<br>01010: Internal trigger input 5 (ITI5)<br>01011: Reserved<br>01100: Internal trigger input 7 (ITI7)<br>01101: Reserved<br>01110: Internal trigger input 9 (ITI9)<br>01111: Internal trigger input 10 (ITI10)<br>10000: Internal trigger input 11 (ITI11)<br>10001: Internal trigger input 12 (ITI12) |

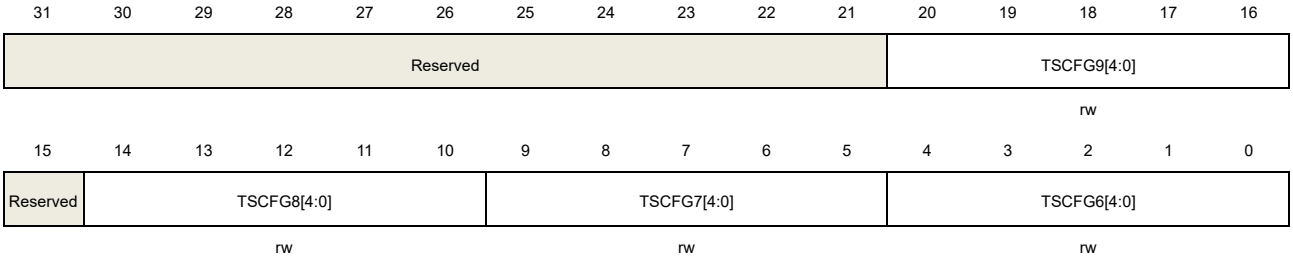
| | | |
|---|---|---|
| | | 10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 15 | Reserved | Must be kept at reset value. |
| 14:10 | TSCFG2[4:0] | Quadrature decoder mode 2 configuration<br>00000: Quadrature decoder mode 2 disable<br>Others: The counter counts on both CI0FE0 and CI1FE1 edges, while the direction depends on each other |
| 9:5 | TSCFG1[4:0] | Quadrature decoder mode 1 configuration<br>00000: Quadrature decoder mode 1 disable<br>Others: The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level |
| 4:0 | TSCFG0[4:0] | Quadrature decoder mode 0 configuration<br>00000: Quadrature decoder mode 0 disable<br>Others: The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level. |

### 1.8.25. TIMERx configuration register 1 (SYSCFG_TIMERxCFG1, x=1, 2, 3, 4, 22, 23, 30, 31)

Address offset: 0x110 for TIMER1
Address offset: 0x11C for TIMER2
Address offset: 0x128 for TIMER3
Address offset: 0x134 for TIMER4
Address offset: 0x158 for TIMER22
Address offset: 0x164 for TIMER23
Address offset: 0x170 for TIMER30
Address offset: 0x17C for TIMER31
Reset value: 0x0000 0000

TSCFG0[4:0], TSCFG1[4:0]..TSCFG9[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | TSCFG9[4:0] | Non-quadrature decoder mode 1 configuration<br>00000: Non-quadrature decoder mode 1 disable<br>Others: The CI0 signal is used as the count pulse( with the CH0P is used to select the counter edge) and the CI1 signal is used as the count direction selection. |
| 15 | Reserved | Must be kept at reset value. |
| 14:10 | TSCFG8[4:0] | Non-quadrature decoder mode 0 configuration<br>00000: Non-quadrature decoder mode 0 disable<br>Others: The CI0 signal is used as the count pulse and CI1 is used as the count selection signal. When CH1P=0, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is high; When CH1P=1, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is low. |
| 9:5 | TSCFG7[4:0] | Restart + event mode configuration<br>The counter is reinitialized and started, the shadow registers are updated on the rising edge of the selected trigger input when these bits are not 0.<br>00000: Restart + event mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: The filtered output of external trigger input (ETIFP)<br>01001: Internal trigger input 4 (ITI4)<br>01010: Internal trigger input 5 (ITI5)<br>01011: Reserved<br>01100: Internal trigger input 7 (ITI7)<br>01101: Reserved<br>01110: Internal trigger input 9 (ITI9)<br>01111: Internal trigger input 10 (ITI10)<br>10000: Internal trigger input 11 (ITI11)<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 4:0 | TSCFG6[4:0] | External clock mode 0 configuration<br>The counter counts on the rising edges of the selected trigger when these bits are not 0.<br>00000: External clock mode 0 disable<br>00001: Internal trigger input 0 (ITI0) |

00010: Internal trigger input 1 (ITI1)
00011: Internal trigger input 2 (ITI2)
00100: Internal trigger input 3 (ITI3)
00101: CI0 edge flag (CI0F_ED)
00110: The filtered output of channel 0 input (CI0FE0)
00111: The filtered output of channel 1 input (CI1FE1)
01000: The filtered output of external trigger input (ETIFP)
01001: Internal trigger input 4 (ITI4)
01010: Internal trigger input 5 (ITI5)
01011: Reserved
01100: Internal trigger input 7 (ITI7)
01101: Reserved
01110: Internal trigger input 9 (ITI9)
01111: Internal trigger input 10 (ITI10)
10000: Internal trigger input 11 (ITI11)
10001: Internal trigger input 12 (ITI12)
10010: Internal trigger input 13 (ITI13)
10011: Internal trigger input 14 (ITI14)
Others: Reserved

### 1.8.26. TIMERx configuration register 2 (SYSCFG_TIMERxCFG2, x=1, 2, 3, 4, 22, 23, 30, 31)

Address offset: 0x114 for TIMER1
Address offset: 0x120 for TIMER2
Address offset: 0x12C for TIMER3
Address offset: 0x138 for TIMER4
Address offset: 0x15C for TIMER22
Address offset: 0x168 for TIMER23
Address offset: 0x174 for TIMER30
Address offset: 0x180 for TIMER31
Reset value: 0x0000 0000

TSCFG0[4:0], TSCFG1[4:0]..TSCFG9[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | TSCFG15[4:0] | | | | |
| | | | | | | | | | | | | | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | TSCFG15[4:0] | Internal trigger input source configuration<br>00000: Reserved<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: Reserved<br>00111: Reserved<br>01000: Reserved<br>01001: Internal trigger input 4 (ITI4)<br>01010: Internal trigger input 5 (ITI5)<br>01011: Reserved<br>01100: Internal trigger input 7 (ITI7)<br>01101: Reserved<br>01110: Internal trigger input 9 (ITI9)<br>01111: Internal trigger input 10 (ITI10)<br>10000: Internal trigger input 11 (ITI11)<br>10001: Internal trigger input 12 (ITI12)<br>10010: Internal trigger input 13 (ITI13)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved<br>**Note:** When TSCFG15[4:0] is used, TSCFGy[4:0](y=0..9) should be zero, otherwise, the ITS trigger cooperate with TSCFGy[4:0], and input source depend on TSCFGy[4:0]. |
| 15:0 | Reserved | Must be kept at reset value. |

### 1.8.27. TIMERx configuration register 0 (SYSCFG_TIMERxCFG0, x=14, 40, 41, 42, 43, 44)

Address offset: 0x148 for TIMER14
Address offset: 0x184 for TIMER40
Address offset: 0x190 for TIMER41
Address offset: 0x19C for TIMER42
Address offset: 0x1A8 for TIMER43
Address offset: 0x1B4 for TIMER44
Reset value: 0x0000 0000

TSCFG3[4:0], TSCFG4[4:0]..TSCFG7[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TSCFG5[4:0] | | | | | TSCFG4[4:0] | | | | | TSCFG3[4:0] | | | | |
| | rw | | | | | rw | | | | | rw | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:26 | TSCFG5[4:0] | Event mode configuration<br>A rising edge of the trigger input enables the counter.<br>00000: Event mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 25:21 | TSCFG4[4:0] | Pause mode configuration<br>The trigger input enables the counter clock when it is high and disables the counter when it is low when these bits are not 0.<br>00000: Pause mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: Reserved<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>10011: Internal trigger input 14 (ITI14) |

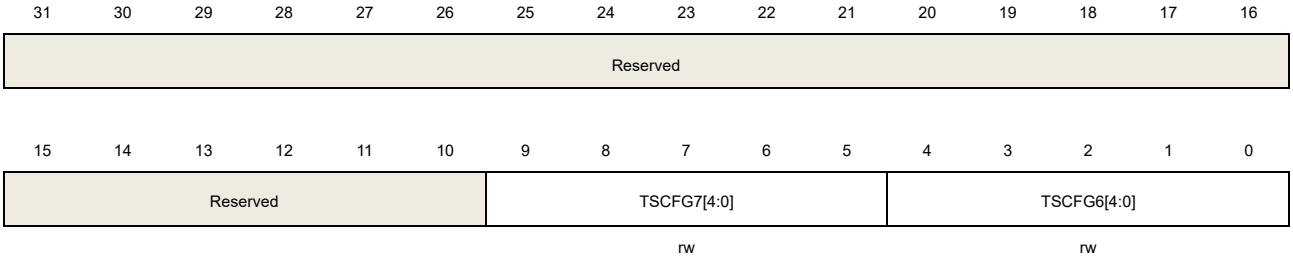
| | | |
|---|---|---|
| | | Others: Reserved |
| 20:16 | TSCFG3[4:0] | Restart mode configuration<br>The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input when these bits are not 0.<br>00000: Restart mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 15:0 | Reserved | Must be kept at reset value. |

### 1.8.28. TIMERx configuration register 1 (SYSCFG_TIMERxCFG1, x=14, 40, 41, 42, 43, 44)

Address offset: 0x14C for TIMER14
Address offset: 0x188 for TIMER40
Address offset: 0x194 for TIMER41
Address offset: 0x1A0 for TIMER42
Address offset: 0x1AC for TIMER43
Address offset: 0x1B8 for TIMER44
Reset value: 0x0000 0000

TSCFG3[4:0], TSCFG4[4:0]..TSCFG7[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9:5 | TSCFG7[4:0] | Restart + event mode configuration<br>The counter is reinitialized and started, the shadow registers are updated on the rising edge of the selected trigger input when these bits are not 0.<br>00000: Restart + event mode disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |
| 4:0 | TSCFG6[4:0] | External clock mode 0 configuration<br>The counter counts on the rising edges of the selected trigger when these bits are not 0.<br>00000: External clock mode 0 disable<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: The filtered output of channel 0 input (CI0FE0)<br>00111: The filtered output of channel 1 input (CI1FE1)<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: The filtered output of multi mode channel 0 input (MCI0FEM0)<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved |

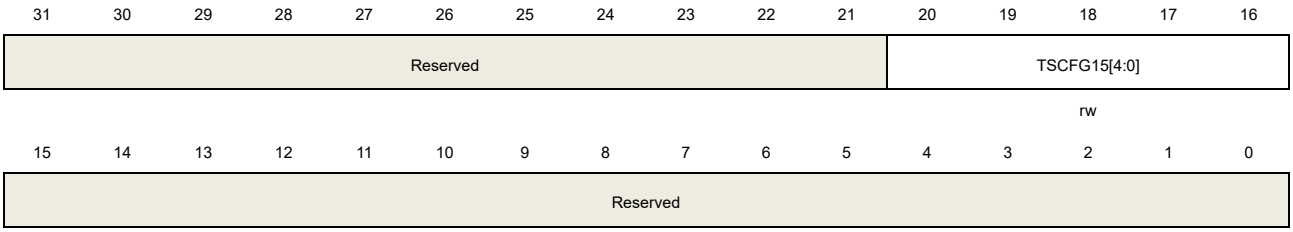
### 1.8.29. TIMERx configuration register 2 (SYSCFG_TIMERxCFG2, x=14, 40, 41, 42, 43, 44)

Address offset: 0x150 for TIMER14
Address offset: 0x18C for TIMER40
Address offset: 0x198 for TIMER41

Address offset: 0x1A4 for TIMER42
Address offset: 0x1B0 for TIMER43
Address offset: 0x1BC for TIMER44
Reset value: 0x0000 0000

TSCFG3[4:0], TSCFG4[4:0]..TSCFG7[4:0] are mutually exclusive and cannot be configured at the same time.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | TSCFG15[4:0] | Internal trigger input source configuration<br>00000: Reserved<br>00001: Internal trigger input 0 (ITI0)<br>00010: Internal trigger input 1 (ITI1)<br>00011: Internal trigger input 2 (ITI2)<br>00100: Internal trigger input 3 (ITI3)<br>00101: CI0 edge flag (CI0F_ED)<br>00110: Reserved<br>00111: Reserved<br>01000: Reserved<br>01001: Reserved<br>01010: Reserved<br>01011: Reserved<br>10011: Internal trigger input 14 (ITI14)<br>Others: Reserved<br>**Note:** When TSCFG15[4:0] is used, TSCFGy[4:0](y = 3..7) should be zero, otherwise, the ITS trigger cooperate with TSCFGy[4:0], and input source depend on TSCFGy[4:0]. |
| 15:0 | Reserved | Must be kept at reset value. |

## 1.8.30. User configuration register (SYSCFG_USERCFG)

Address offset: 0x300
Reset value: 0x0000 00XX

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | BOOT_MODE[2:0] | | | Reserved | | BOR_TH[1:0] | |
| | | | | | | | | | r | | | | | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6:4 | BOOT_MODE[2:0] | Boot mode<br>These bits indicate the mode of BOOT.<br>000: BOOT from SRAM (ITCM/DTCM/RAM shared/AXI SRAM)<br>001: BOOT from Security<br>010: BOOT_SYS (BootLoader)<br>011: BOOT_USER (User flash OSPI0/1)<br>Others: Reserved |
| 3:2 | Reserved | Must be kept at reset value. |
| 1:0 | BOR_TH[1:0] | BOR threshold status bits<br>00: No BOR function<br>01: BOR threshold value 1<br>10: BOR threshold value 2<br>11: BOR threshold value 3 |

# 1.9. AXI interconnect registers

AXI base address: 0x5100 0000

## 1.9.1. AXI peripheral ID4 register (AXI_PERIPH_ID4)

Address offset: 0x1FD0
Reset value: 0x0000 0004

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | 4KBCNT[3:0] | | | | JEP106CCODE[3:0] | | | |
| | | | | | | | | r | | | | r | | | |

| Bits | Fields | Descriptions |
|---|---|---|

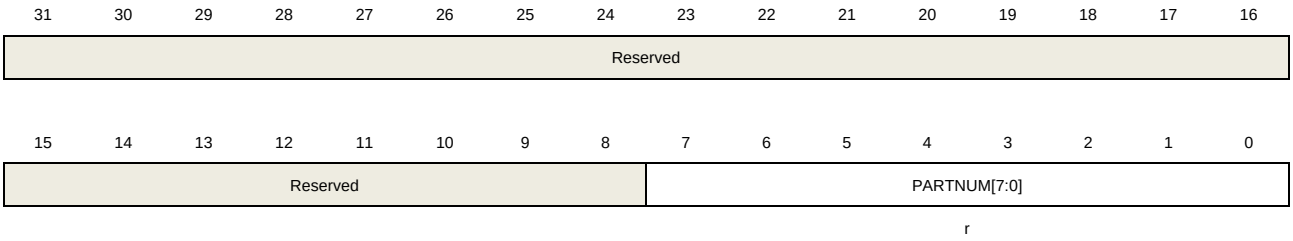
| | | |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | 4KBCNT[3:0] | 4KB count |
| 3:0 | JEP106CCODE[3:0] | JEP106 continuation code |

### 1.9.2. AXI peripheral ID0 register (AXI_PERIPH_ID0)

Address offset: 0x1FE0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



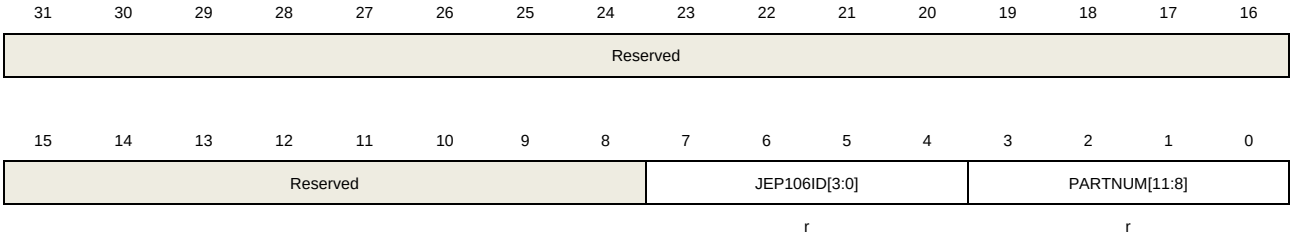

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | PARTNUM[7:0] | Part number[7:0] |

### 1.9.3. AXI peripheral ID1 register (AXI_PERIPH_ID1)

Address offset: 0x1FE4

Reset value: 0x0000 00B4

This register has to be accessed by word (32-bit).



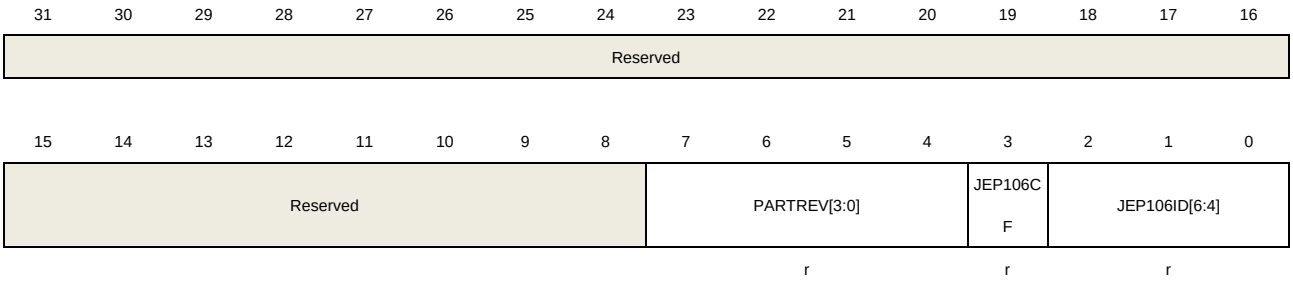

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | JEP106ID[3:0] | JEP106 Identity[3:0] |
| 3:0 | PARTNUM[11:8] | Part number[11:8] |

### 1.9.4. AXI peripheral ID2 register (AXI_PERIPH_ID2)

Address offset: 0x1FE8

Reset value: 0x0000 002B

This register has to be accessed by word (32-bit).



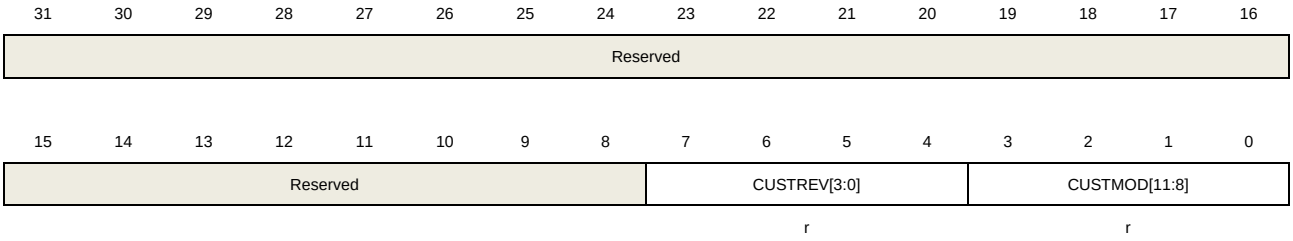

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | PARTREV[3:0] | Part revision |
| 3 | JEP106CF | JEP106 code flag |
| 2:0 | JEP106ID[6:4] | Part number[6:4] |

## 1.9.5. AXI peripheral ID3 register (AXI_PERIPH_ID3)

Address offset: 0x1FEC

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | CUSTREV[3:0] | Customer version |
| 3:0 | CUSTMOD[11:8] | Customer modification |

## 1.9.6. AXI componet ID0 register (AXI_COMP_ID0)

Address offset: 0x1FF0

Reset value: 0x0000 000D

This register has to be accessed by word (32-bit).



| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | PREAMB[7:0] | | | | | | | |
| | | | | | | | | r | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | PREAMB[7:0] | Preamble bits |

### 1.9.7. AXI componet ID1 register (AXI_COMP_ID1)

Address offset: 0x1FF4

Reset value: 0x0000 00F0

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | CLASS[3:0] | | | | PREAMB[11:8] | | | |
| | | | | | | | | r | | | | r | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:4 | CLASS[3:0] | Component class |
| 3:0 | PARTNUM[11:8] | Preamble bits |

### 1.9.8. AXI componet ID2 register (AXI_COMP_ID2)

Address offset: 0x1FF8

Reset value: 0x0000 0005

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | PREAMB[19:12] | | | | | | | |
| | | | | | | | | r | | | | | | | |

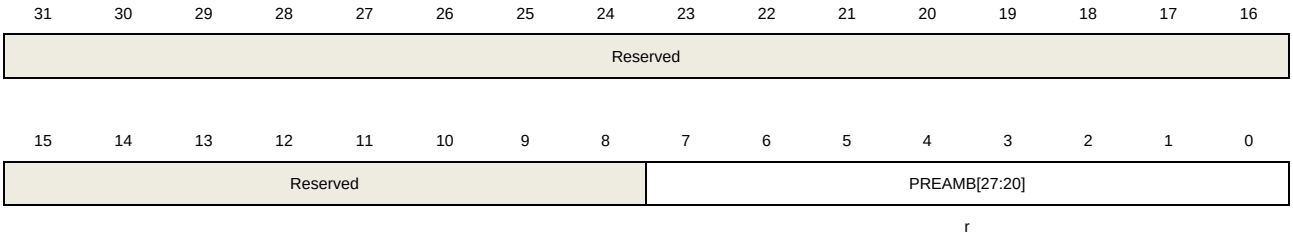
| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | PREAMB[19:12] | Preamble bits |

### 1.9.9. AXI componet ID3 register (AXI_COMP_ID3)

Address offset: 0x1FFC
Reset value: 0x0000 00B1

This register has to be accessed by word (32-bit).

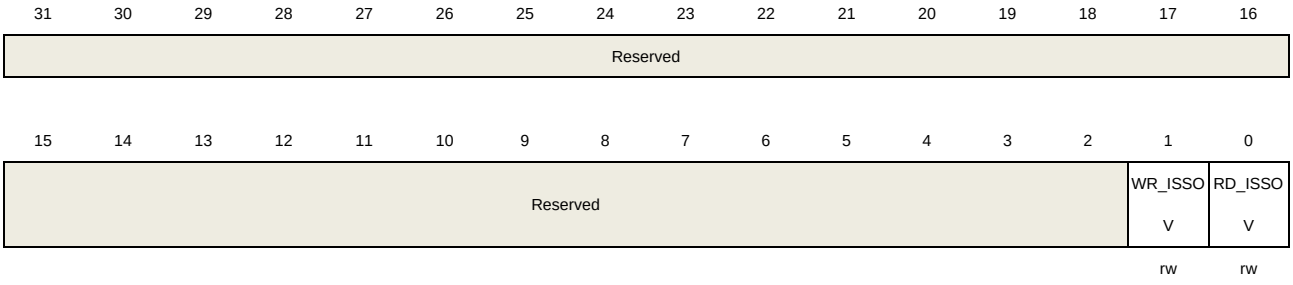



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | PREAMB[27:20] | Preamble bits |

### 1.9.10. AXI Master Port x bus matrix issuing functionality control register (AXI_MPxBM_ISS_CTL)

Address offset: 0x2008 + 0x1000 * x, where x = 0 to 7
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



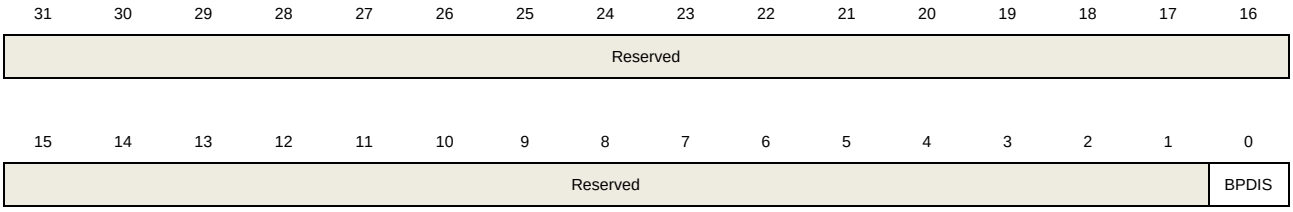

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WR_ISSOV | Override target write issuing function<br>0: Normal issuing function<br>1: The bus matrix write issuing capability is set to 1 |
| 0 | RD_ISSOV | Override target read issuing function<br>0: Normal issuing function |

1: The bus matrix read issuing capability is set to 1

### 1.9.11. AXI Master Port x bus matrix functionality control register (AXI_MPxBM_CTL)

Address offset: 0x2024 + 0x1000 * x, where x = 0, 1, 6 and 7
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

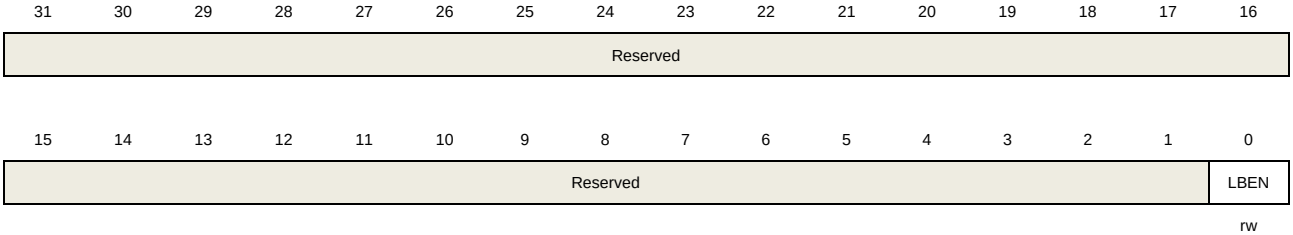



| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | BPDIS | Beats packing function disable configure<br>0: Normal operation<br>1: Disable beats packing function |

### 1.9.12. AXI Master Port x long burst functionality control register (AXI_MPx_LB_CTL)

Address offset: 0x202C + 0x1000 * x, where x = 0 and 1
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

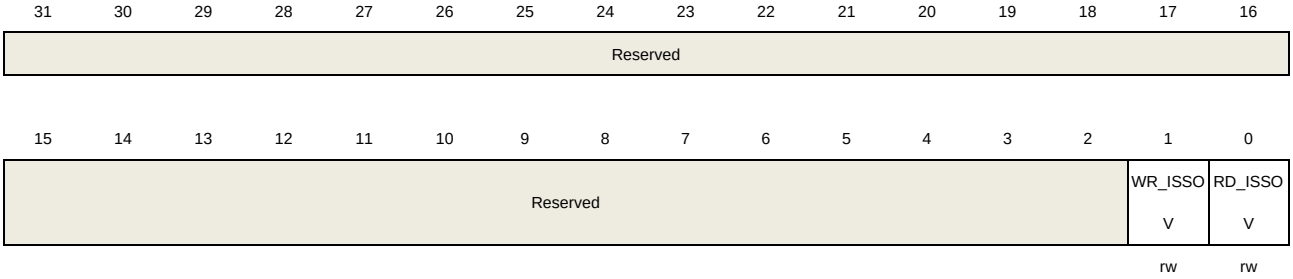



| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | LBEN | Control long burst function<br>0: Disable long burst generated at the output of the ASIB<br>1: Enable long burst generated at the output of the ASIB |

### 1.9.13. AXI Master Port x issuing functionality control register (AXI_MPx_ISS_CTL)

Address offset: 0x2108 + 0x1000 * x, where x = 0, 1, 2, and 7
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

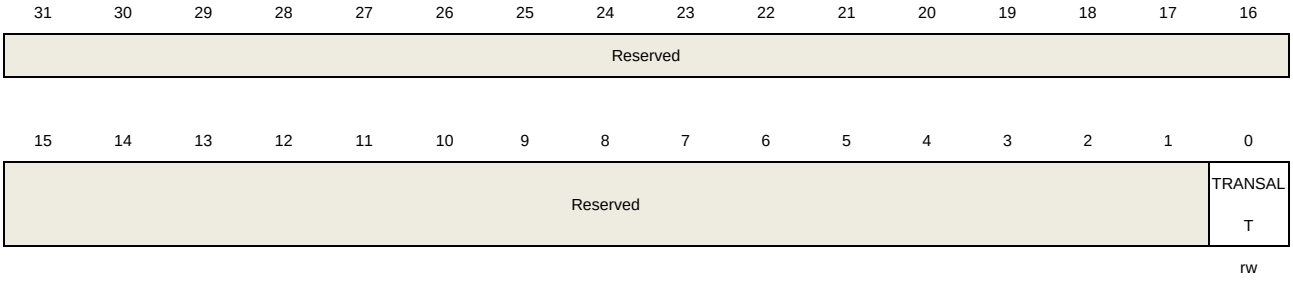



| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WR_ISSOV | Override AMIB write issuing function<br>0: Normal issuing function<br>1: The AMIB write issuing capability is forced set to 1 |
| 0 | RD_ISSOV | Override AMIB read issuing function<br>0: Normal issuing function<br>1: The AMIB read issuing capability is forced set to 1 |

### 1.9.14. AXI Slave Port x functionality control register (AXI_SPx_CTL)

Address offset: 0x42024 + 0x1000 * x, where x = 0 and 2
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



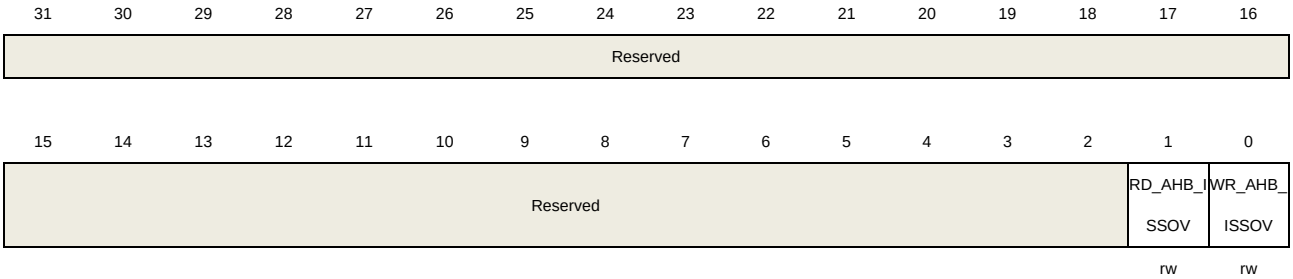

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | TRANSALT | Transaction alteration configure<br>0: Normal operation |

1: Transaction unaltered where allowed

### 1.9.15. AXI Slave Port x AHB issuing functionality control register (AXI_SPx_AHBISS_CTL)

Address offset: 0x42028 + 0x1000 * x, where x = 0 and 2
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

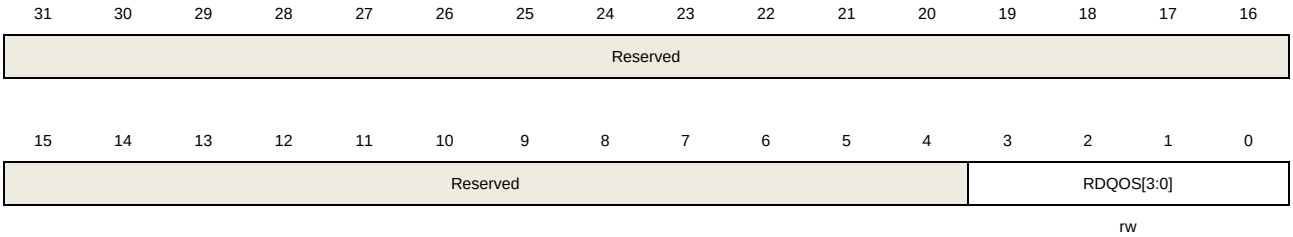



| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | RD_AHB_ISSOV | Converts AHB-Lite read transaction to single beat AXI transaction function<br>0: Override disabled<br>1: Override enabled |
| 0 | WR_AHB_ISSOV | Converts AHB-Lite write transaction to single beat AXI transaction function<br>0: Override disabled<br>1: Override enabled |

### 1.9.16. AXI Slave Port x read QOS control register (AXI_SPx_RDQOS_CTL)

Address offset: 0x42100 + 0x1000 * x, where x = 0 to 5
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3:0 | RDQOS[3:0] | Read channel QoS configure |

0000: Lowest priority

…

1111: Highest priority

### 1.9.17. AXI Slave Port x write QOS control register (AXI_SPx_WRQOS_CTL)

Address offset: 0x42104 + 0x1000 * x, where x = 0 to 5

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | WRQOS[3:0] | | | |
| | | | | | | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3:0 | WRQOS[3:0] | Write channel QoS configure<br>0000: Lowest priority<br>…<br>1111: Highest priority |

### 1.9.18. AXI Slave Port x issuing functionality control register (AXI_SPx_ISS_CTL)

Address offset: 0x42108 + 0x1000 * x, where x = 0 to 5

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | WR_ISSOV | RD_ISSOV |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WR_ISSOV | Override ASIB write issuing function<br>0: Normal issuing function |

| | | |
|---|---|---|
| | | 1: The ASIB write issuing capability is forced set to 1 |
| 0 | RD_ISSOV | Override ASIB read issuing function<br>0: Normal issuing function<br>1: The ASIB read issuing capability is forced set to 1 |

## 1.10. Device electronic signature

The device electronic signature contains memory density information and the 96-bit unique device ID. The 96-bit unique device ID is unique for each device. It can be used as serial numbers, or part of security keys, etc.

### 1.10.1. Memory density information

Base address: 0x1FF0 F7E0

The value is factory programmed and can never be altered by user.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLASH_DENSITY[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRAM_DENSITY[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | FLASH_DENSITY [15:0] | Flash memory density<br>The value indicates the Flash memory density of the device in Kbytes.<br>Example: 0x0020 indicates 32 Kbytes. |
| 15:0 | SRAM_DENSITY [15:0] | SRAM density<br>The value indicates the on-chip SRAM density of the device in Kbytes.<br>Example: 0x0008 indicates 8 Kbytes. |

### 1.10.2. Unique device ID (96 bits)

Base address: 0x1FF0 F7E8

The value is factory programmed and can never be altered by user.

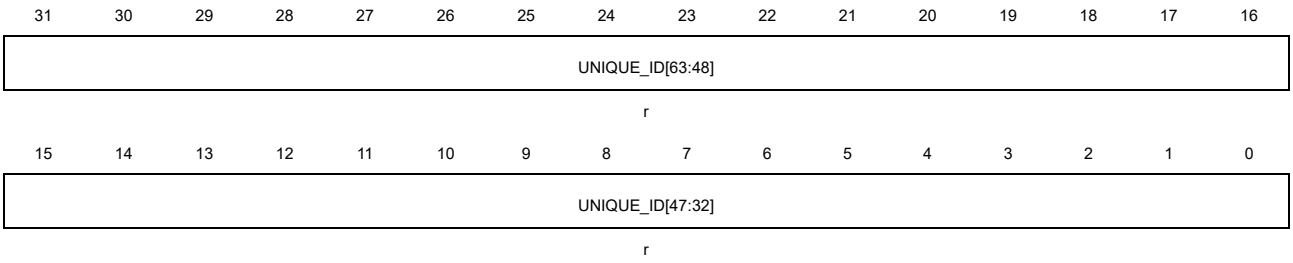
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UNIQUE_ID[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | UNIQUE_ID[31:0] | Unique device ID |

Base address: 0x1FF0 F7EC

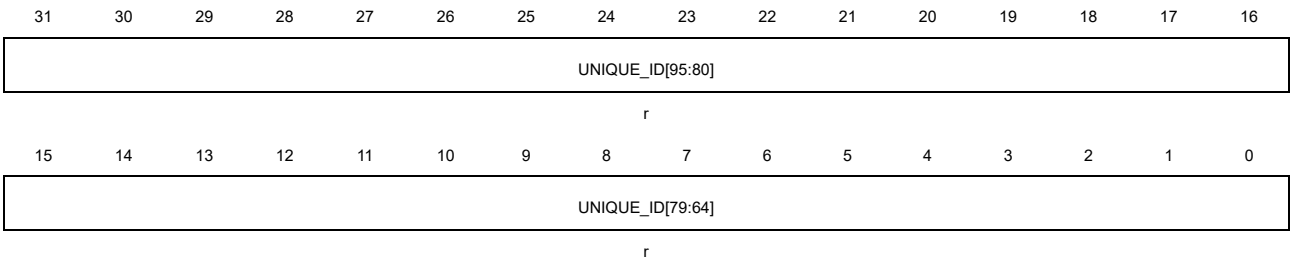
The value is factory programmed and can never be altered by user.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | UNIQUE_ID[63:32] | Unique device ID |

Base address: 0x1FF0 F7F0

The value is factory programmed and can never be altered by user.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | UNIQUE_ID[95:64] | Unique device ID |

# 2. RAM ECC monitor unit (RAMECCMU)

The GD32H73x_75x device features two RAM ECC monitor units (RAMECCMU) in Region 0 and Region1 separately. It provides a method to verify ECC status for applications and execute error handling when errors occur.

## 2.1. Characteristics

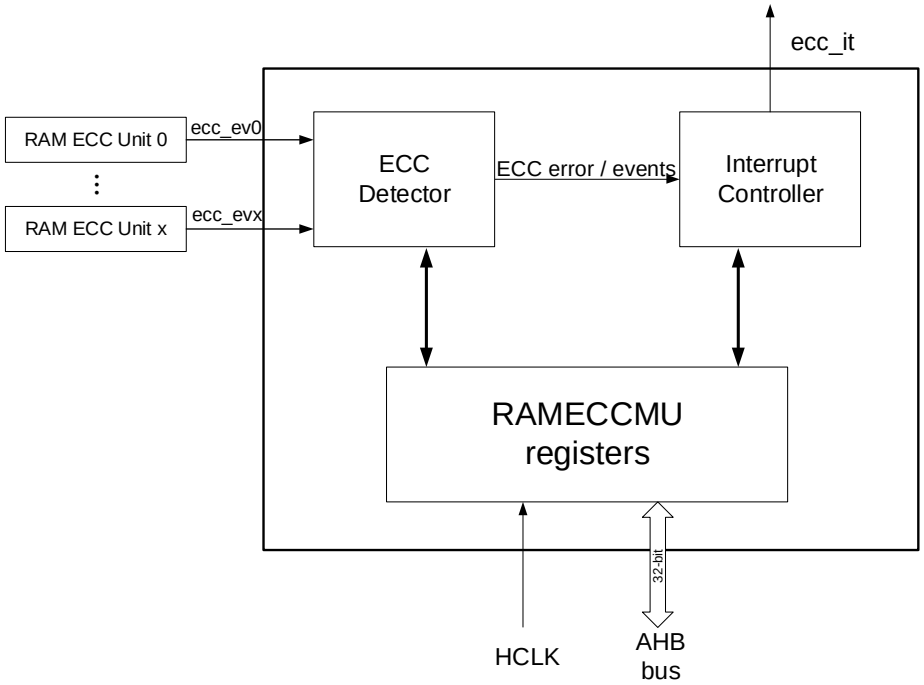
The main functions of RAMECCMU are the following:

- RAM ECC monitor for each Region
- Identification for RAM failing address/data

## 2.2. Function overview

GD32H73x_75x features two RAMECC monitor units mounted on AHB3 in Region 0 and AHB2 in Region 1 separately. The block architechture of RAMECCMU is shown as ***Figure 2-1. Block architecture of RAMECCMU***.

**Figure 2-1. Block architecture of RAMECCMU**




There are two RAMECC monitor units in GD32H73x_75x series which are described in ***Table 2-1. RAMECC monitor x unit for Region 0 (x=0..4)*** and ***Table 2-2. RAMECC monitor x unit for Region 1 (x=0..2)***.

**Table 2-1. RAMECC monitor x unit for Region 0 (x=0..4)**

| RAMECC monitor | RAMECC monitor status |
|---|---|

| number | |
|---|---|
| 0 | AXI SRAM ECC |
| 1 | ITCM-RAM ECC |
| 2 | DTCM-RAM ECC(D0TCM) |
| 3 | DTCM-RAM ECC(D1TCM) |
| 4 | RAM shared(ITCM/DTCM/AXI SRAM) ECC |

**Table 2-2. RAMECC monitor x unit for Region 1 (x=0..2)**

| RAMECC monitor number | RAMECC monitor status |
|---|---|
| 0 | SRAM0 ECC |
| 1 | SRAM1 ECC |
| 2 | Backup RAM(BKPSRAM) ECC |

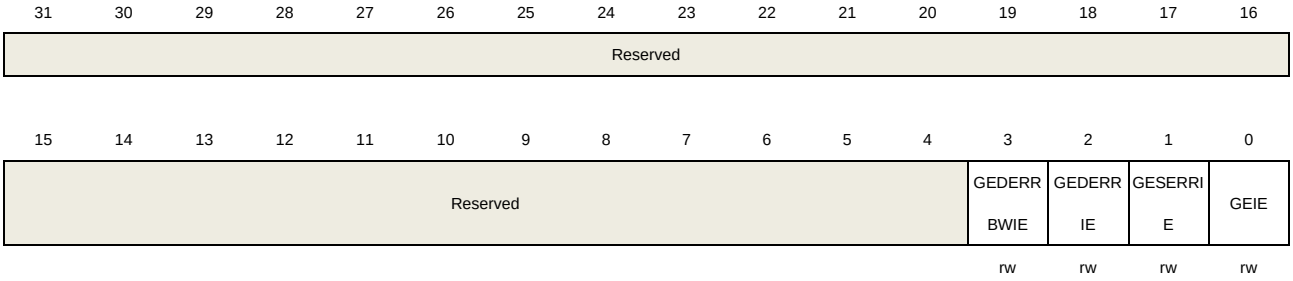
## 2.3. Register definition

RAMECCMU Region 0 base address: 0x5200 9000
RAMECCMU Region 1 base address: 0x4802 3000

### 2.3.1. RAMECCMU global interruput register (RAMECCMU_INT)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | GEDERRBWIE | Global ECC double error on byte write interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when ECC double detection error occurs during a byte write operation to RAM |
| 2 | GEDERRIE | Global ECC double error interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when ECC double detection error occurs during a read operation from RAM |
| 1 | GESERRIE | Global ECC single error interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when ECC single error occurs during a read operation from RAM |
| 0 | GEIE | Global ECC interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when one of GEDERRBWIE, GEDERRIE or GESERRIE error occurs |

### 2.3.2. RAMECCMU monitor x control register (RAMECCMU_MxCTL)

Address offset: 0x20 * (x+1), (x is ECC monitoring number, x=0..4 for Region 0, while x=0..2 for Region 1)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | ECCERRLATEN | ECCDERRBWIE | ECCDERRIE | ECCSERRIE | Reserved | |
| | | | | | | | | | | rw | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5 | ECCERRLATEN | ECC error latching enable<br>0: No error context are latched to their respective registers when ECC error occurs<br>1: Error context are latched to their respective registers when ECC error occurs |
| 4 | ECCDERRBWIE | ECC double error on byte write interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when ECC double error occurs on byte write operation to RAM |
| 3 | ECCDERRIE | ECC double error interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when ECC double error occurs on read operation from RAM |
| 2 | ECCSERRIE | ECC single error interrupt enable<br>0: No interrupt generated<br>1: Interrupt generated when ECC single error occurs on read operation from RAM |
| 1:0 | Reserved | Must be kept at reset value. |

### 2.3.3. RAMECCMU monitor x status register (RAMECCMU_MxSTAT)

Address offset: 0x24 + 0x20 * x, (x= ECC monitoring number, x=0..4 for Region 0, while x=0..2 for Region 1)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | ECCDERRBWDF | ECCDERRDF | ECCSERRDCF |
| | | | | | | | | | | | | | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | ECCDERRBWDF | ECC double error on byte write detected flag<br>This bit is set by hardware and cleared by writing 0.<br>0: Double error is detected during read when ECCDERRDF is set<br>1: Double error detected during unaligned write |
| 1 | ECCDERRDF | ECC double error detected flag<br>0: No double error detected<br>1: Double error detected |
| 0 | ECCSERRDCF | ECC single error detected and corrected flag<br>0: No error detected and corrected<br>1: Error detected and corrected |

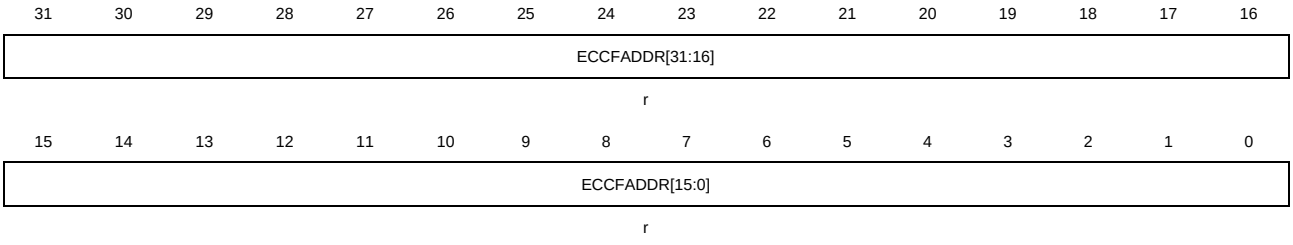
### 2.3.4. RAMECCMU monitor x failing address register (RAMECCMU_MxFADDR)

Address offset: 0x28 + 0x20 * x, (x= ECC monitoring number, x=0..4 for Region 0, while x=0..2 for Region 1)

Reset value: 0x2400 0000 for AXI SRAM
0x0000 0000 for ITCM
0x2000 0000 for D0TCM
0x2000 0004 for D1TCM
0x2408 0000 for RAM shared(ITCM/DTCM/AXI SRAM)

0x3000 0000 for SRAM0
0x3000 4000 for SRAM1
0x3880 0000 for BKPSRAM

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ECCFADDR[31:0] | ECC error failing address<br>This register contains the address which generated ECC error when error occurs. |

### 2.3.5. RAMECCMU monitor x failing data low register (RAMECCMU_MxFDL)

Address offset: 0x2C + 0x20 * x, (x= ECC monitoring number, x=0..4 for Region 0, while x=0..2 for Region 1)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCFDL[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCFDL[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ECCFDL[31:0] | ECC failing data low bits<br>This register contains the LSB of data which generated ECC error when error occurs or the full memory word for 32-bit word SRAM. |

### 2.3.6. RAMECCMU monitor x failing data high register (RAMECCMU_MxFDH)

Address offset: 0x30 + 0x20 * x, (x= ECC monitoring number, x=0..4 for Region 0, while x=0..2 for Region 1)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCFDH[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCFDH[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ECCFDH[31:0] | ECC failing data high bits (64-bit)<br>This register contains the MSB of data which generated ECC error when error occurs. |

### 2.3.7. RAMECCMU monitor x failing ECC error code register (RAMECCMU_MxFECODE)

Address offset: 0x34 + 0x20 * x, (x= ECC monitoring number, x=0..4 for Region 0, while x=0..2 for Region 1)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCFECODE[31:16] | | | | | | | | | | | | | | | |

r

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCFECODE[15:0] | | | | | | | | | | | | | | | |

r

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ECCFECODE[31:0] | ECC failing error code<br>This register contains the index where the bit error occurs and the ECC code. |

# 3. Flash memory controller (FMC)

## 3.1. Overview

The flash memory controller, FMC, provides all the necessary functions for the on-chip flash memory. It also provides sector erase, mass erase, program operations for flash memory.

## 3.2. Characteristics

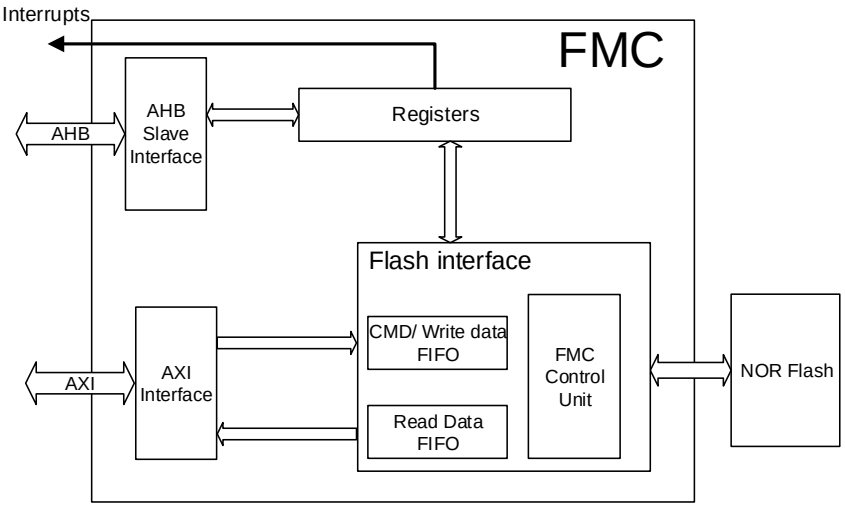
- Up to 3840KB of on-chip flash memory for instruction and data.
- Double-word/word/half-word/byte reading operation.
- Double-word/word programming, sector erase and mass erase operation.
- Option bytes are uploaded to the option byte control registers on every system reset
- Flash security protection to prevent illegal code/data access.
- Sector erase/program protection to prevent unexpected operation.
- Provides one execute-only dedicated code read protection ( DCRP ) area.
- Provides one secure user area which is accessible only in secure mode.

## 3.3. Function overview

### 3.3.1. Flash memory architecture

FMC supports 64-bit AXI interface to access code/data and 32-bit AHB slave interface to access register. *Figure 3-1. FMC block diagram* shows the details.

**Figure 3-1. FMC block diagram**




The AXI interface of FMC can support read and write operation at the same time. The AXI slave interface supports the following access types:

- Flash memory single read operations supporting 1,2,4,8-byte.
- Flash memory incrementing burst read operations supporting size 1,2,4,8-byte; burst length up to 128 bytes.
- Flash memory wrapped burst read operations supporting size 8-byte with 1,2,4,8-beat.
- Single write accesses supporting size 4,8-byte.
- Burst write accesses supporting size 8-byte with most 32-beat, and burst write with 4-beat is most efficient.
- If using the burst write which supporting size 8-byte with multi-beat, only MDMA can be used instead of DMA, otherwise unpredictable errors will occur;
- Write address must align with size.
- Address can not across 4K boundary in one burst read/write operation.
- All operations exclusive above description will generate a bus error, read operation return zero, write operation is ignored.

The AHB slave interface supports 32-bit word accesses.

The flash memory consists of 3840KB main flash organized into 960 sectors with 4KB and 64KB information block. Each sector can be erased individually.

*Table 3-1. GD32H73x_75x base address and size for flash memory* shows the details of flash organization

**Table 3-1. GD32H73x_75x base address and size for flash memory**

| Block | Name | Address range | Size(bytes) |
|---|---|---|---|
| Main flash block | Sector 0 | 0x0800 0000 - 0x0800 0FFF | 4KB |
| | Sector 1 | 0x0800 1000 - 0x0800 1FFF | 4KB |
| | Sector 2 | 0x0800 2000 - 0x0800 2FFF | 4KB |
| | . . . | . . . | . . . |
| | Sector 959 | 0x083B F000 - 0x083B FFFF | 4KB |
| Information block | Bootloader | 0x1FF0 0000 - 0x1FF0 FFFF | 64KB |

### 3.3.2. Read operations

The flash can be addressed directly as a common memory space. Any instruction fetch and the data access from the flash are through AXI interface from the CPU.

#### FMC internal real-time decryption (RTDEC) function

FMC internal RTDEC function means that when reading data from Flash, it can be decrypted in real time according to the EFUSE module's configuration of AES key. (Data written to Flash is encrypted already). There have on-time decrypt function when AESEN bit in EFUSE_USER_CTL register is set. This is implement by hardware on-time and invisible by software. The AES key use AESKEYx[31:0] bits in EFUSE_AES_KEYx to set. The initialization vector AES_IV[127:0] = AESIV[95:0] || 12'b0 || ReadAddress[23:4]. User can

modify the high 96 bits of the initial vector (AESIV[95:0]) by modifying the FMC_AESIVx_MDF register. The AESIV[95:0] is formed with [AESIV2, AESIV1,AESIV0]. When the initial vector needs to be modified, user must write the FMC_ASIV0_MDF, FMC_ASIV1_MDF, and FMC_ASIV2_MDF registers in sequence. After the FMC_AESIV2_MDF register is written, the value in the FMC_ASIV0/1/2_MDF registers will be updated to the AES IV area.

The following steps show the AESIV[95:0] modification sequence.

1. Unlock the FMC_CTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no Flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
3. Write the initialization vector AES_IV[32:63] into FMC_AESIV0_MDF, AES_IV[64:95] into FMC_AESIV1_MDF, and AES_IV[96:127] into FMC_AESIV2_MDF register in sequence;
4. Once the FMC_AESIV2_MDF register is written, the value in the FMC_AESIV0/1/2_MDF register will be automatically updated to the AES IV area.
5. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT register.
6. Launch a system reset to start option bytes loading.
7. Read from FMC_AESIVx_EFT register and verify the values if required.

When the operation is executed successfully, the ENDF in FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set.

#### NO-RTDEC function

Area can be configured without RTDEC function by FMC_NODEC, even if AESEN bit in EFUSE_USER_CTL register is set.

### 3.3.3. Unlock the FMC_CTL and FMC_OBCTL register

After reset, the FMC_CTL register is not accessible in write mode, and the LK bit in FMC_CTL register is 1. An unlocking sequence consists of two write operations to the FMC_KEY register to open the access to the FMC_CTL register. The two write operations are writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register. After the two write operations, the LK bit in FMC_CTL register is reset to 0 by hardware. The software can lock the FMC_CTL again by setting the LK bit in FMC_CTL register to 1. Any wrong operations to the FMC_KEY, set the LK bit to 1, and lock FMC_CTL register, and lead to a bus error. The FMC_OBCTL registers are still protected even the FMC_CTL is unlocked. The unlocking sequence is two write operations, which are writing 0x08192A3B and 0x4C5D6E7F to FMC_OBKEY register. After the two write operations, the OBLK bit in FMC_OBCTL register is reset to 0 by hardware. The software can lock the FMC_OBCTL again by setting the OBLK bit in FMC_OBCTL register to 1.

### 3.3.4. Sector erase

The FMC provides a sector erase function which is used to initialize the contents of a main flash block sector to a high state. Each sector can be erased independently without affecting the contents of other sectors. The following steps show the access sequence of the registers for a sector erase operation.

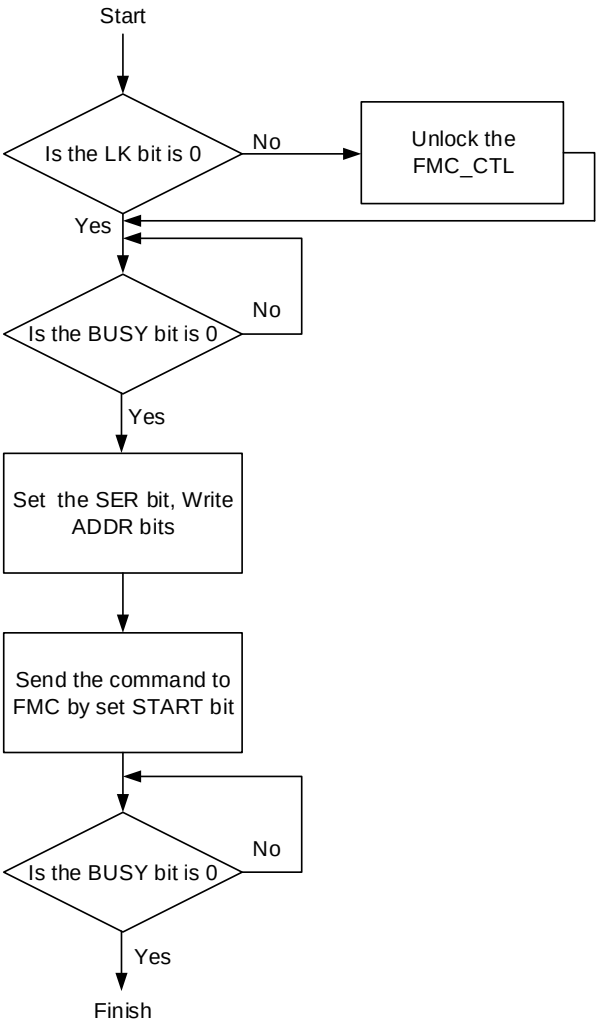
1. Unlock the FMC_CTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
3. Set the SER bit in FMC_CTL register.
4. Write the sector address ADDR to FMC_ADDR register.
5. Send the sector erase command to the FMC by setting the START bit in FMC_CTL register.
6. Wait until all the operations have finished by checking the value of the BUSY bit in FMC_STAT register.
7. Read and verify the sector if required.

If a main flash sector contain DCRP, secure user or erase/program protected data, the sector erase operation is stopped and the WPERR bit is set in the FMC_STAT register.

If mass erase and sector erase is request at the same time, the mass erase operation will replace the sector erase operation.

When the operation is executed successfully, the ENDF in FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Note that a correct target sector address must be confirmed. Or the software may run out of control if the target erase sector is being used to fetch codes or to access data. The FMC will not provide any notification when this occurs. Additionally, the sector erase operation will be ignored on erase/program protected sectors. In this condition, a erase/program protection error interrupt will be triggered by the FMC if the WPERRIE bit in the FMC_CTL register is set. The software can check the WPERR bit in the FMC_STAT register to detect this condition in the interrupt handler. ***Figure 3-2. Proccess of sector erase operation*** shows the sector erase operation flow.

**Figure 3-2. Proccess of sector erase operation**




**Note:** When programming, erasing, especially the mass erasing, abnormal power off or reset should be avoided as far as possible, otherwise unpredictable consequences may occur. If there is a risk of power off and reset, avoid using protection-removed mass erasing.

**Note:** When modifying option bytes, do not power off or reset, otherwise it will damage the flash data and have unpredictable consequences that may force the chip into secure mode.

**Note:** When programming and erasing data in the main flash block, do not power off or reset, otherwise the data in the current sector may be damaged. After the next power on, user need to erase the damaged sector before continuing to use it.

### 3.3.5. Mass erase

#### Typical mass erase

The FMC provides a typical mass erase function which is used to erase all sectors except sectors contain secure user or protected data. This erase can affect by setting the MER bit to 1 in the FMC_CTL register. The following steps show the typical mass erase register access sequence.

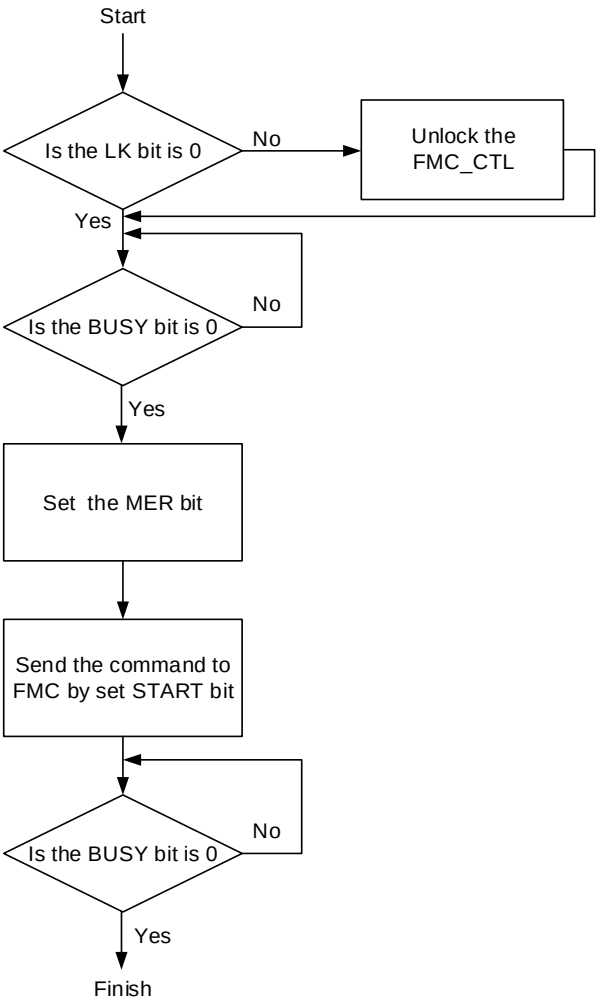
1. Unlock the FMC_CTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
3. Set the MER bit in the FMC_CTL register to enable mass erase operation.
4. Send the mass erase command to the FMC by setting the START bit in FMC_CTL register.
5. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT register.
6. Read and verify the flash memory if required.

If mass erase and sector erase is request at the same time, the mass erase operation will replace the sector erase operation.

When the operation is executed successfully, the ENDF in FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Since all flash data except sectors contain secure user or protected data will be modified to a value of 0xFFFF_FFFF, the mass erase operation can be implemented using a program that runs in SRAM or by using the debugging tool that accesses the FMC registers directly.

***Figure 3-3. Process of typical mass erase operation*** shows the typical mass erase operation flow.

**Figure 3-3. Process of typical mass erase operation**




**Note:** When programming, erasing, especially the mass erasing, abnormal power off or reset should be avoided as far as possible, otherwise unpredictable consequences may occur. If there is a risk of power off and reset, avoid using protection-removed mass erasing.

**Note:** When modifying option bytes, do not power off or reset, otherwise it will damage the flash data and have unpredictable consequences that may force the chip into secure mode.

**Note:** When programming and erasing data in the main flash block, do not power off or reset, otherwise the data in the current sector may be damaged. After the next power on, user need to erase the damaged sector before continuing to use it.

### Protection-removed mass erase

The FMC provides a protection-removed mass erase function which is used to erase all sectors include sectors contain secure user or protected data. This erase can affect by setting the MER bit to 1 in the FMC_CTL register. The following steps show the protection-removed mass erase register access sequence.

1. Unlock the FMC_OBCTL register if necessary.
2. If a DCRP area exists, set DCRP_EREN bit in FMC_DCRPADDR_MDF or in FMC_DCRPADDR_EFT register. And set DCRP end address less than DCRP start

address by programming DCRP_AREA_END < DCRP_AREA_START to FMC_DCRPADDR_MDF register.

3. If a secure user area exists, set SCR_EREN bit in FMC_SCRADDR_MDF or in FMC_SCRADDR_EFT register. And set secure user area end address less than secure user area start address by programming SCR_AREA_END < SCR_AREA_START to FMC_SCRADDR_MDF register.
4. Set all WP bits in FMC_WP_MDF register if any erase/program protected sector exists.
5. Unlock the FMC_CTL register if necessary.
6. Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
7. Set the MER bit in the FMC_CTL register to enable mass erase operation.
8. Send the mass erase command to the FMC by setting the START bit in FMC_CTL register.
9. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT register. The operation has erased the whole main flash memory including the sectors containing DCRP-protected and/or secure user data. And the correspond option bytes modified automatically performed for disable all the protections.
10. Read and verify the flash memory if required.

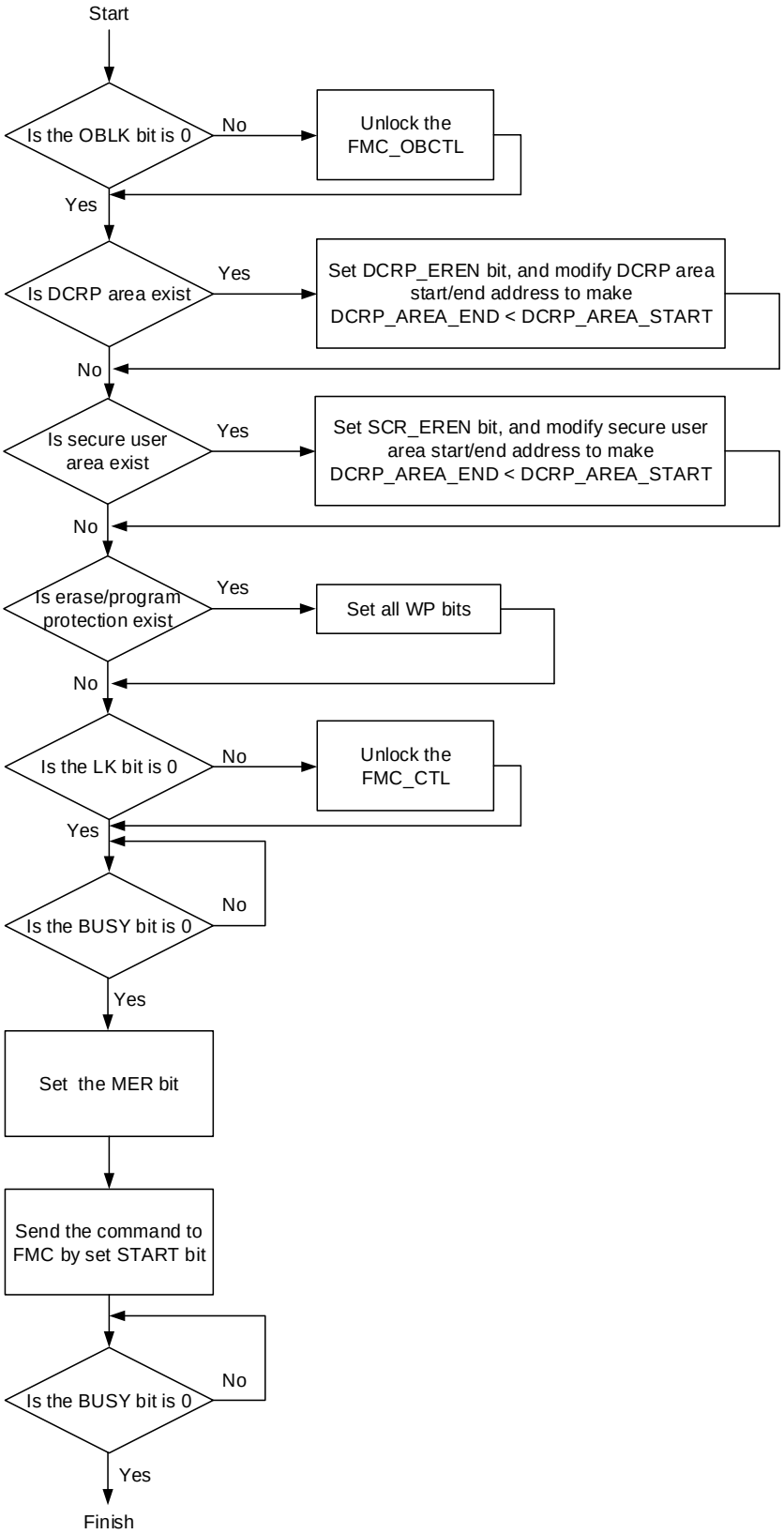
**Note:** (1) In the above sequence, except for the option bytes indicated above, other option bytes cannot be modified. (2) Only if all the conditions in steps 2, 3, and 4 are satisfied, the protection-removed mass erase will be performed. But if any of the steps are not met, only the typical mass erase will be performed, and no error will be raised.

If mass erase and sector erase is request at the same time, the mass erase operation will replace the sector erase operation.

When the operation is executed successfully, the ENDF in FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Since all flash data (include sectors contain secure user or protected data) will be modified to a value of 0xFFFF_FFFF, the mass erase operation can be implemented using a program that runs in SRAM or by using the debugging tool that accesses the FMC registers directly.

*Figure 3-4. Process of protection-removed mass erase operation* shows the protection-removed mass erase operation flow.

**Figure 3-4. Process of protection-removed mass erase operation**




**Note:** When programming, erasing, especially the mass erasing, abnormal power off or reset should be avoided as far as possible, otherwise unpredictable consequences may occur. If there is a risk of power off and reset, avoid using protection-removed mass erasing.

**Note:** When modifying option bytes, do not power off or reset, otherwise it will damage the flash data and have unpredictable consequences that may force the chip into secure mode.

**Note:** When programming and erasing data in the main flash block, do not power off or reset, otherwise the data in the current sector may be damaged. After the next power on, user need to erase the damaged sector before continuing to use it.

### 3.3.6. Main flash Programming

The FMC provides a 64-bit/32-bit programming function by AXI interface which is used to modify the main flash block contents. The following steps show the register access sequence of the word programming operation.

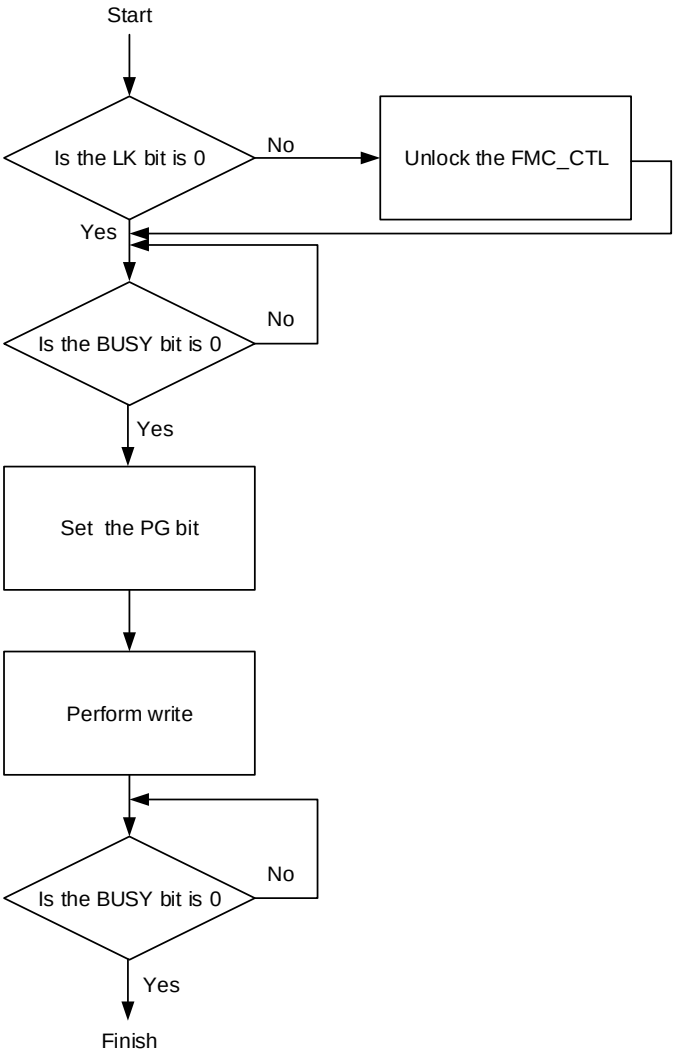
Unlock the FMC_CTL register if necessary.
Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
Set the PG bit in FMC_CTL register.
Write the data to be programed with desired absolute address (0x08XX XXXX).
Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT register.
Read and verify the Flash memory if required.

When the operation is executed successfully, the ENDF in FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set. Note that the PG bit must be set before the double-word/word programming operation, or else PGSERR bit in the FMC_STAT register will be set, and an interrupt will be triggered by FMC if the PGSERRIE bit in the FMC_CTL register is set. Additionally, the program operation will be ignored on erase/program protected sectors and WPERR bit in FMC_STAT is set, and an interrupt will be triggered by FMC if the WPERRIE bit in the FMC_CTL register is set. The software can check the PGSERR or WPERR bit in the FMC_STAT register to detect which condition occurred in the interrupt handler. ***Figure 3-5. Proccess of program operation*** displays the word programming operation flow.

User can enable check whether the programming area is all 0xFF before programming by set PGCHEN bit in FMC_CTL register. If the programming area is not all 0xFF, the PGSERR in the FMC_STAT register will be set.

If PGCHEN is set and burst program is enable, FMC will check the flash data according to the burst size. If it is checked that the data is not all 0xFF, this beat program is failed and PGSERR bit in FMC_STAT register is set. But other beats can be written normally.

**Figure 3-5. Proccess of program operation**




**Note:** When programming, erasing, especially the mass erasing, abnormal power off or reset should be avoided as far as possible, otherwise unpredictable consequences may occur. If there is a risk of power off and reset, avoid using protection-removed mass erasing.

**Note:** When modifying option bytes, do not power off or reset, otherwise it will damage the flash data and have unpredictable consequences that may force the chip into secure mode.

**Note:** When programming and erasing data in the main flash block, do not power off or reset, otherwise the data in the current sector may be damaged. After the next power on, user need to erase the damaged sector before continuing to use it.

## 3.3.7. Option bytes

### Option bytes description

FMC provides two groups of option byte registers:

"_EFT" registers group ( read-only )

These registers store the effective value of the option bytes. After a option byte modification

operation, system reset or wakeup from standby mode, the registers values will be automatically loaded from non-volatile memory.

"_MDF" registers group ( read/write )

These registers store the modification of the option bytes. When user wants to modify the option byte, the desired option byte value needs to be written to the register group.

The option bytes block is reloaded to "_EFT" registers after each system reset, and the option bytes take effect. The option bytes description is shown in the ***Table 3-2. Option byte***.The option bytes are configured according to the requirements of the application.

**Table 3-2. Option byte**

| Name | Factory value | Register map |
|---|---|---|
| [29]: IOSPDOPEN | 0x0 | FMC_OBSTAT0_EFT / FMC_OBSTAT0_MDF |
| [24]: DTCM1ECCEN | 0x1 | |
| [23]: DTCM0ECCEN | 0x1 | |
| [22]: ITCMECCEN | 0x1 | |
| [21]: SCR | 0x0 | |
| [18]: FWDGSPD_STDBY | 0x1 | |
| [17]: FWDGSPD_DPSLP | 0x1 | |
| [15:8]: SPC[7:0] | 0xAA | |
| [7]: nRST_STDBY | 0x1 | |
| [6]: nRST_DPSLP | 0x1 | |
| [4]: nWDG_HW | 0x1 | |
| [3:2]: BOR_TH[1:0] | 0x0 | |
| [31]: DCRP_EREN | 0x0 | FMC_DCRPADDR_EFT / FMC_DCRPADDR_MDF |
| [26:16]: DCRP_AREA_END[10:0] | 0x0 | |
| [10:0]: DCRP_AREA_START[10:0] | 0x0FF | |
| [31]: SCR_EREN | 0x0 | FMC_SCRADDR_EFT / FMC_SCRADDR_MDF |
| [26:16]: SCR_AREA_END[10:0] | 0x000 | |
| [10:0]: SCR_AREA_START[10:0] | 0x0FF | |
| [21:0]: WP[21:0] | 0x3FFFFFFF | FMC_WP_EFT / FMC_WP_MDF |
| [31:16]: BOOT_ADDR1[15:0] | 0x1FF0 | FMC_BTADDR_EFT / FMC_BTADDR_MDF |
| [15:0]: BOOT_ADDR0[15:0] | 0x0800 | |
| [31:16]: DATA[15:0] | 0x0 | FMC_OBSTAT1_EFT / FMC_OBSTAT1_MDF |
| [7:4]: DTCM_SZ_SHRRAM[3:0] | 0x8 | |
| [3:0]: ITCM_SZ_SHRRAM[3:0] | 0x7 | |

### Option bytes modify

The following steps show the erase sequence.

1. Unlock the FMC_OBCTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no Flash memory operation

is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.

3. Write the desired option byte value in FMC_XXX_MDF.
4. Send the option bytes change command to the FMC by setting the OBSTART bit in FMC_OBCTL register.
5. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC_STAT register.
6. Launch a system reset to start option bytes loading.
7. Read from FMC_XXX_EFT register and verify the option bytes if required.

**Note:** "XXX" includes OBSTAT0, DCRPADDR, SCRADDR, WP, BTADDR or OBSTAT1.

When the operation is executed successfully, the ENDF in FMC_STAT register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC_CTL register is set.

If OBLK bit in the FMC_OBCTL register is set, the _MDF register is forbidden to modify. When the OBSTART bit is set to 1, FMC will compare the effective value (_EFT) with a new value (_MDF) to check whether have option byte needs to be changed.

**Note:** When modifying option bytes, do not power off or reset, otherwise it will damage the flash data and have unpredictable consequences that may force the chip into secure mode.

**Note:** When programming and erasing data in the main flash block, do not power off or reset, otherwise the data in the current sector may be damaged. After the next power on, user need to erase the damaged sector before continuing to use it.

### Option bytes modify rules

Some of the option bytes modification must respect specific rules. If one of the rules below is not met, OBMERR flag is set and modification operation is stopped. If all rules below are met, FMC starts to modify option byte and updates the _EFT option byte register value.

- Security protection (SPC)

If SPC is set to level high, any option byte modification operation is forbidden. Therefore, if the user application attempts to demote the SPC level, OBMERR flag is set and all programmed modification will be ignored.

- Erase/program protection (WP)

These option bytes manage erase/program protection in the FMC_WP_EFT register. If the SPC protection level is not level high, they can be modified without any restriction.

- DCRP area size (DCRP_AREA_START and DCRP_AREA_END)

The size of DCRP area can be increased without any restriction, but it must contain the original DCRP area, which means DCRP_AREA_START cannot be larger and DCRP_AREA_END cannot be smaller.

When removing the DCRP area, the DCRP_EREN bit (in the FMC_DCRPADDR_EFT or FMC_DCRPADDR_MDF register) must be set to 1, and a SPC level low to no protection demotion or a protection-removed mass erase must be performed simultaneously.

When reducing the size of DCRP area, the DCRP_EREN bit ( in the FMC_DCRPADDR_EFT or FMC_DCRPADDR_MDF register ) must be set to 1, and a SPC level low to no protection demotion must be performed simultaneously.

**Note:** When removing or reducing the size of DCRP area by SPC level low to no protection demotion, if the DCRP_EREN bits ( in the FMC_DCRPADDR_EFT and FMC_DCRPADDR_MDF registers ) are both 0, an option byte modification error ( OBMERR ) will be generated.

- DCRP_EREN

If this option bit is set, the content of the corresponding DCRP area is erased during SPC level low to no protection demotion or a protection-removed mass erase. Otherwise will be preserved.

The DCRP_EREN bit can be set without any restriction. The DCRP_EREN bit can be reset only when the SPC level low to no protection demotion or a protection-removed mass erase is requested simultaneously. Otherwise OBMERR flag is set.

- Secure mode (SCR)

If there is no valid secure user area and no valid DCRP area, SCR option bit can be reset freely. If a valid DCRP area or a valid secure user area exist, the only way to reset SCR option bit is: to perform a SPC level low to no protection demotion when DCRP_EREN ( in FMC_DCRPADDR_EFT or FMC_DCRPADDR_MDF register ) is set to 1 and SCR_EREN ( in FMC_SCRADDR_EFT or FMC_SCRADDR_MDF register ) is set to 1. Otherwise OBMERR flag is set.

**Note:** It is recommended to make both SCR_AREA_START > SCR_AREA_END and DCRP_AREA_START > DCRP_AREA_END programmed when reset the SCR option bit during an SPC level low to no protection demotion.

- Secure user area size (SCR_AREA_START and SCR_AREA_END)

GD secure libraries run on user secure applications or devices can modify them without any restrictions.

For user non-secure applications, when removing the secure user area, the SCR_EREN bit (in the FMC_SCRADDR_EFT or FMC_SCRADDR_MDF register) must be set to 1, and a SPC level low to no protection demotion or a protection-removed mass erase must be performed simultaneously.

**Note:** When removing or reducing the size of SCR area by SPC level low to no protection demotion, if the SCR_EREN bits ( in the FMC_DCRPADDR_EFT and FMC_DCRPADDR_MDF registers ) are both 0, an option byte modification error ( OBMERR ) will be generated.

- SCR_EREN

If SCR_EREN bit is set, the content of the corresponding secure user area is erased during a SPC level low to no protection demotion or a protection-removed mass erase. Otherwise

the content is preserved.

The SCR_EREN bit can be set without any restriction. The SCR_EREN bits can be reset only when the SPC level low to no protection demotion or a protection-removed mass erase is requested simultaneously. Otherwise OBMERR flag is set.

### 3.3.8. Sector erase/program protection

The FMC provides sector erase/program protection functions to prevent inadvertent operations on the Flash memory. The sector erase or program will not be accepted by the FMC on protected sectors. If the sector erase or program command is sent to the FMC on a protected sector, the WPERR bit in the FMC_STAT register will then be set by the FMC. Note that the WPERR also set when sector erase while MER set or sector address not valid. The sector protection function can be individually enabled by configuring the WP[21:0] bit field to 0 in the option bytes.

**Table 3-3. WP bit for sectors protected**

| WP bit | Sectors protected |
|---|---|
| WP[0] | Sector 0~Sector 15 |
| WP[1] | Sector 16~Sector 31 |
| .<br>.<br>. | .<br>.<br>. |
| WP[14] | Sector 224~ Sector 239 |
| WP[15] | Sector 240~ Sector 255 |
| WP[16] | Sector 256~Sector 383 |
| WP[17] | Sector 384~Sector 511 |
| .<br>.<br>. | .<br>.<br>. |
| WP[20] | Sector 768~ Sector 895 |
| WP[21] | Sector 896~Sector 959 |

**Note:** For WP [x] (x=0 … 15), 1 bit corresponds to 16 sectors, ranging from sector (x * 16) to sector (x * 16 + 15). For WP [x] (x=17 … 21), 1 bit corresponds to 128 sectors, from sector (x * 128-1792) ~ sector (x * 128-1665). For WP[21], 1 bit corresponds to 64 sectors, from sector 896 ~ sector 959.

Erase/program protected sectors cannot either be deleted or programmed. Therefore, if a sector is erase/program protected, mass erase cannot be performed unless a protection-removed mass erase or a SPC level low to no protection demotion is performed.

When SPC is set to level high, WP[21:0] cannot be modified, else WP[21:0] can be modified without any restrictions

**Note:** DCRP or secure user areas are erase/program protected.

### 3.3.9. Security protection

The FMC provides a security protection function to prevent illegal code/data access on the Flash memory. This function is useful for protecting the software or firmware from illegal users. Security protection is global. Not only flash memory but also other secured regions are protected. Other secured regions include backup SRAM (BKPSRAM), RTC backup registers and encrypted regions protected by real-time decryption (RTDEC) (RTDEC only for GD32H737_757_759).There are 3 levels for protecting.

#### No protection

When setting SPC[7:0] bits in FMC_OBSTAT0_EFT to 0xAA, no protection performed. The main flash and option bytes block are accessible by all operations. And access to other secured regions is also allowed.

#### Protection level low

When SPC_H in eFuse is 0, as long as setting SPC_L to 1 or SPC[7:0] option bits to any value except 0xAA or 0xCC, protection level low performed. The main flash can only be accessed by user code. In debug mode, boot from SRAM or boot from boot loader mode, all operations to main flash or other secured regions (e.g. backup SRAM) is forbidden. If a read operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, a read protection error (RPERR) with bus error will be generated. If a program/erase operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, the WPERR bit in FMC_STAT register will be set. At protection level low, option bytes block are accessible by all operations. If program back to no protection level by setting SPC byte to 0xAA, a erase events flash will be performed.

The SPC level low to no protection demotion will cause erase events below:

- For the main flash block sectors not belonging to DCRP / secure user area / WP, these sectors will be mass erased.
- Erase / program protection sectors will be erased.
- For DCRP area, if both DCRP_EREN bits in FMC_DCRPADDR_EFT and FMC_DCRPADDR_MDF are 0, the sectors belonging DCRP areas will not be erased. Otherwise DCRP areas will be erased ( overlapped or not with secure-only area ).
- For secure user area, if both SCR_EREN bits in FMC_SCRADDR_EFT and FMC_SCRADDR_MDF are 0, the sectors belonging secure-only areas will not be erased. Otherwise secure-only areas will be erased ( overlapped or not with DCRP area ).
- If at least one DCRP_EREN in FMC_DCRPADDR_EFT or FMC_DCRPADDR_MDF is 1, and at least one SCR_EREN in FMC_SCRADDR_EFT or FMC_SCRADDR_MDF is 1, the whole main flash block will be erased.
- The other secured regions ( such as backup SRAM ) are erased.

**Note:** the sector protections ( DCRP, secure user area) are removed only if the protected sector boundaries are modified by the user application.

### Protection level high

When SPC_H in the eFuse is 1 or the SPC option byte is set to 0xCC, protection level high performed. When this level is programmed, debug mode, boot from SRAM or boot from boot loader mode are disabled. The main flash block is accessible by all operations from user code. The user option bits can be read but can not be modified. And accesses to the other secured areas are also allowed. The SPC byte cannot be reprogrammed. So, if protection level high is programmed, it cannot move back to protection level low or no protection level.

The debug port is always disabled when level high is active. Therefore, it is impossible to debug and analyze the defective parts with level high security protection. If security protection level high is set while the debugger is still connected, apply a power on reset.

User can configure the SPC protection level refer to ***Table 3-4. SPC protection level configuration.***

**Table 3-4. SPC protection level configuration**

| EFUSE_USER_CTL register | | FMC_OBSTAT0_MDF register | FMC_OBSTAT0_EFT register | SPC level |
|---|---|---|---|---|
| SPC_H | SPC_L | SPC[7:0] | SPC[7:0] | |
| 1 | 0 or 1 | Any value | = 0xCC | Level high |
| 0 or 1 | 0 or 1 | 0xCC | = 0xCC | Level high |
| 0 | 1 | Any value except 0xCC | = 0xFF | Level low |
| 0 | 0 | Any value except 0xCC or 0xAA | = SPC[7:0] value in FMC_OBSTAT0_MDF register | Level low |
| 0 | 0 | 0xAA | = 0xAA | No protection |

**Note:** (1) If SPC level setting in efuse macro is different from SPC level setting in option byte, the SPC level depend on the higher level of the two settings. And SPC[7:0] bits in FMC_OBSTAT0_EFT register reflect the effective SPC level. (2)If SPC_L bit in efuse macro is set, SPC level low to no protection demotion is forbidden. Otherwise OBMERR bit will be set in FMC_STAT register.

## 3.3.10. Dedicated code read protection area

In the main flash block, FMC can define executable-only areas, allowing only instruction transactions from the system, but not data access.

**Note:** Users need to compile their native code accordingly, using the execution-only option when apply executable-only area function.

One DCRP area can be defined by setting the DCRP_AREA_END[10:0] and DCRP_AREA_START[10:0] option bytes with a granularity of 4 Kbytes. This means that the actual DCRP area size is defined by:

- [( DCRP_AREA_END[10:0] - DCRP_AREA_START[10:0] ) + 1] x 4Kbytes

As an example, to set a DCRP area on sector 0 to 15, the option bytes should be set as

follows:

- DCRP_AREA_START[10:0] = 0x000,
- DCRP_AREA_END[10:0] = 0x00F,

So the DCRP size is:

- [( DCRP_AREA_END[10:0] - DCRP_AREA_START[10:0] ) + 1] x 4Kbytes = 64 Kbytes.

The minimum DCRP area is 8Kbytes. The maximum DCRP area is the entire main flash block, configured by setting the same value to the start and end addresses of the DCRP area.

When executing code in this area, the debug events will be ignored. Only CPU can access DCRP area, using only instruction fetch transactions. In all other cases, access to the DCRP area is illegal. For example, read operations will return 0 and a RPERR flag in FMC_STAT register is set, and write operations will be ignored and a WPERR in FMC_STAT register is set.

A valid DCRP area is erase-protected. Cannot erase sectors which located in this area. If a valid DCRP area is setting, mass erase cannot be performed unless erase is performed during the SPC level low to no protection demotion or protection-removed mass erase.

Only CPU can modify the DCRP area definition bits and DCRP_EREN bit. If DCRP area is valid(not empty), during SPC level low to no protection demotion, if in FMC_DCRPADDR_EFT and FMC_DCRPADDR_MDF, the DCRP_EREN bits are both set to 0, the contents of the DCRP area will not be erased, otherwise the contents will be erased.

Besides the option bytes, the DCRP area can also be configured by modifying the MCU reserved parameter in efuse macro with granularity of 32KB bytes.

If the DCRPLK bit in MCU reserved parameter in efuse macro is 1 and DCRP_AREA_END[7:0] and DCRP_AREA_START[7:0] in efuse macro is not all 0, DCRP area is defined by the DCRP_AREA_END[7:0] and DCRP_AREA_START[7:0] bits in the Efuse MCU reserved parameter. Otherwise, DCRP area is defined by the DCRP_AREA_END[10:0] and DCRP_AREA_START[10:0] bits in option bytes. For more details, please refer to ***Table 3-5. DCRP area configuration***.

**Table 3-5. DCRP area configuration**

| DCRP area configure in Option Byte | DCRP area configure in Efuse | DCRPLK configure in Efuse | Effective DCRP area |
|---|---|---|---|
| Area1 | Start and end addresses are both 0. | 0 | Area1(2) |
| | Start and end addresses are both 0. | 1 | Area1(2) |
| | Area2(1) | 0 | Area1(2) |
| | Area2(1) | 1 | Area2 |

**Note:** (1) The start and end addresses of Area2 are not both 0. (2) If the DCRP area is defined by Efuse, the value of DCRP_AREA_START[10:0] in the FMC_DCRPADDR_MDF and FMC_DCRPADDR_EFT registers is { DCRP_AREA_START[7:0] in Efuse, 3'b0 }, and the

value of DCRP_AREA_END[10:0] is { DCRP_AREA_END[7:0] in Efuse, 3'b111 }.

If the DCRP area is decided by Efuse. The DCRP_AREA_END[10:0] bits or DCRP_AREA_START[10:0] bits in FMC_DCRPADDR_MDF register can not be modified. So the DCRP area is fixed. The protection-removed erase will not work.

**Note:** (1) The DCRP configuration priority of efuse is higher than the flash option byte, so in the product, if configure DCRP area by the option byte, the start and end address of the DCRP area in efuse should be both set to 0, and the DCRPLK bit in efuse should be set to 1, otherwise there may be vulnerability in the DCRP area. (2) If the user has security considerations, please configure the DCRP area by efuse, otherwise there may be security risks.

### 3.3.11. Secure user area

In the main flash block, FMC can define secure user areas which can only be accessed when the CPU executes a secure application.

Secure user areas can isolate secure code from application non-secure code. Secure user areas can be used to protect a custom secure boot library, firmware update code, or a third party secure library.

One secure user area can be defined by setting the SCR_AREA_END and SCR_AREA_START option bytes with a granularity of 4 Kbytes. This means that the actual secure user area size is defined by:

- [( SCR_AREA_END[10:0] - SCR_AREA_START[10:0] ) + 1] x 4Kbytes

As an example, to set a secure user area on sector 0, the option bytes should be set as follows:

- SCR_AREA_START[10:0] = 0x000,
- SCR_AREA_END[10:0] = 0x00F,

So the secure user size is:

[( SCR_AREA_END[10:0] - SCR_AREA_START[10:0] ) + 1] x 4Kbytes = 64 Kbytes.

SCR_AREA_END[10:0] and SCR_AREA_START[10:0] option bytes can be modified by CPUs running secure libraries or application secure code. But for user non-secure applications, these option bytes can be modified only during SPC level low to no protection demotion or a protection-removed mass erase.

The minimum secure user area is 8Kbytes. The maximum secure user area is the entire main flash block, configured by setting the same value to the start and end addresses of the secure user area.

When executing code in this area, the debug events will be ignored. Only CPU can access it by executing GD secure library or user secure application. In all other cases, accessing the secure user area is illegal. For example, read operations will return 0 and a RSERR flag in

FMC_STAT register is set, and write operations will be ignored and a WPERR in FMC_STAT register is set.

A valid secure user area is erase-protected. Cannot erase sectors located in this area.

Unless the application is executed from a valid secure user area, it is impossible to erase the sector located in the area. If a valid secure user area is defined, mass erase cannot be performed unless erase is performed during the SPC level low to no protection demotion or protection-removed mass erase.

Only CPU can modify the secure user area definition bits and SCR_EREN bit. If secure user area is valid ( not empty ), during the SPC level low to no protection demotion, if the SCR_EREN bit in FMC_SCRADDR_EFT register and the SCR_EREN bit in FMC_SCRADDR_MDF register are both set to 0, the contents of the secure user area will not be erased, otherwise the contents will be erased.

Besides the option bytes, the secure user area can also be configured by modifying the User control parameter in efuse macro with granularity of 32KB bytes.

If the SCRLK bit in User control parameter in efuse macro is 1 and SCR_AREA_END[7:0] and SCR_AREA_START[7:0] in Efuse macro is not all 0, secure user area is defined by the SCR_AREA_END[7:0] and SCR_AREA_START[7:0] bits in the Efuse user control parameter. Otherwise, secure user area is defined by the SCR_AREA_END[10:0] and SCR_AREA_START[10:0] bits in option bytes. For more details, please refer to ***Table 3-6. Secure user area configuration***.

**Table 3-6. Secure user area configuration**

| Secure user area configure in Option Byte | Secure user area configure in Efuse | SCRLK configure in Efuse | Effective Secure user area |
|---|---|---|---|
| Area1 | Start and end addresses are both 0. | 0 | Area1(2) |
| | Start and end addresses are both 0. | 1 | Area1(2) |
| | Area2(1) | 0 | Area1(2) |
| | Area2(1) | 1 | Area2 |

**Note:** (1) The start and end addresses of Area2 are not both 0. (2) If the securie user area is defined by Efuse, the value of SCR_AREA_START[10:0] in the FMC_SCRADDR_MDF and FMC_SCRADDR_EFT registers is {SCR_AREA_START[7:0] in Efuse, 3'b0}, and the value of SCR_AREA_END[10:0] is {SCR_AREA_END[7:0] in Efuse, 3'b111}.

If the secure user area is decided by Efuse. The SCR_AREA_END[10:0] bits or SCR_AREA_START[10:0] bits in FMC_SCRADDR_MDF register can not be modified. So the Secure user area is fixed. The protection-removed erase will not work.

**Note:** (1) The secure configuration priority of efuse is higher than the flash option byte, so in the product, if configure the secure user area by the option byte, the start and end address of the secure user area in efuse should be both set to 0, and the SCRLK bit in efuse should be set to 1, otherwise there may be vulnerability in the secure user area. (2) If the user has

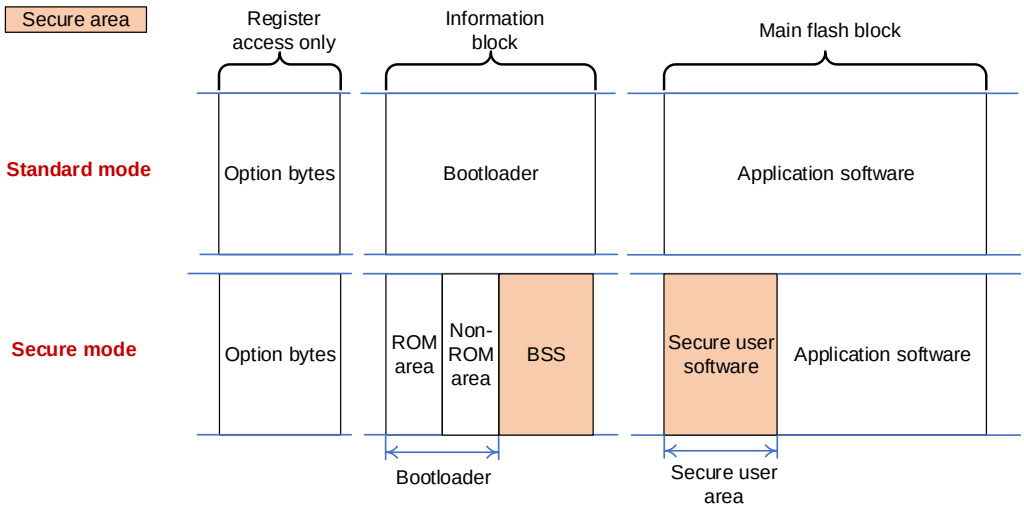
security considerations, please configure the secure user area by efuse, otherwise there may be security risks.

### 3.3.12. Secure mode

Security should be performed for some sensitive programs to avoid potentially malware attacks. For example, licensed firmware update software requires highly protection because it processes confidential data ( such as encryption keys ) that cannot get by other processes.

Secure areas with limited access is provided. In this area, secure services can be built that can be executed before any user application. These secure areas and their included software can be accessed only in secure mode. ***Figure 3-6. Memory architecture in standard mode and secure mode*** shows the details of the area.

**Figure 3-6. Memory architecture in standard mode and secure mode**




Secure user area is accessed once after reset, the area code is hidden after execution. Basic security service is Gigadevice software to configure secure services. Secure user software is located in secure user area and executed once after reset. Secure user software can be used to implement secure boot and licensed firmware update.

Secure mode and secure user area can be configured by option bytes or efuse. User can set the secure user area in the option byte by BSS to make the secure code and data be configured in the secure user area.

In secure mode, secure firmware store in information block to support boot. Secure user software is secure application code, data, or algorithms stored in main flash block.

If secure user area is not set, MCU will jump to the requested boot address which set by the BOOT_ADDR0[15:0] option bits in FMC_BTADDR_EFT register.

In secure mode, regardless of the startup configuration ( BOOT pins and boot addresses ), MCU will be forced to boot from secure ROM area.

Secure mode is enabled as long as one of the SCR bits in the Option byte or Efuse is set to 1. At this time, the SCR bit of the FMC_OBSTAT0_EFT register is 1. And after the SCR bit is enabled, a system reset is required to activate secure mode.

Standard mode can be returned when the SCR bit in option byte is 1 and the SCR bit in efuse is 0. To return to standard mode, the secure area and DCRP area need to be removed before or at the same time as the SCR option bit is cleared. For more details, please refer to the modify rules of the relevant option bytes.

**Note:** If user has security considerations, please configure the secure mode by efuse, otherwise there may be security risks.

### 3.3.13. Basic security service

BSS provides the secure area setting function and secure area exiting function.

- Secure area setting function

Secure area setting function is provided by Gigadevice to perform the initialization of secure user area. In standard mode, user can directly call function (resetAndInitializeSecureAreas) to set the secure area, and other basic security services are not allowed to access.

The description of ***Table 3-7. Function resetAndInitializeSecureAreas*** is shown as below:

**Table 3-7. Function resetAndInitializeSecureAreas**

| | |
|---|---|
| **Function name** | resetAndInitializeSecureAreas |
| **Function prototype** | void resetAndInitializeSecureAreas(BSS_secure_area_struct area); |
| **Function descriptions** | Set the range of the secure user area based on the SCR_AREA_START and SCR_AREA_END option bytes. |
| **Precondition** | - |
| **The called functions** | - |
| **Input parameter{in}** | |
| **area** | secure user area start address and end address |
| **Output parameter{out}** | |
| **-** | - |
| **Return value** | |
| **-** | - |

**Note:** After the function is completed, a system reset is generated. This function is available only when the secure user area is first set up. User must ensure that the correct secure programs is exist in the target secure area to make it can exit to the standard programs, otherwise it will cause chip scrap.

- Secure area exiting function

Gigadevice provides a function ( exitSecureArea ) to exit from secure user software and jump to user application. It can close secure user area to ensure that the content in secure user area is no longer accessed.

The description of ***Table 3-8. Function exitSecureArea*** is shown as below:

**Table 3-8. Function exitSecureArea**

| | |
|---|---|
| **Function name** | exitSecureArea |
| **Function prototype** | void exitSecureArea(unsigned int vectors, unsigned int jtagState); |
| **Function descriptions** | exit from the secure user area and jump to the user application. |
| **Precondition** | - |
| **The called functions** | - |
| **Input parameter{in}** | |
| **vectors** | address of application vector to jump after exit secure user area. |
| **Input parameter{in}** | |
| jtag_state | status of the JTAG after exit secure user area. |
| *BSS_EXIT_SCR_JTAG_ENABLE* | enable the JTAG after exiting |
| *BSS_EXIT_SCR_JTAG_DISABLE* | Disable the JTAG after exiting |
| **Output parameter{out}** | |
| **-** | - |
| **Return value** | |
| **-** | - |

**Note:** After the function is completed, no system reset is generated. For security reasons, users should disable cache before calling this function in a secure user area.

## 3.3.14. Error description

### Erase/program protection error (WPERR)

The following erase operations will be rejected and the WPERR bit in the FMC_STAT register will be set:

- Erase sectors in the valid DCRP area;
- Erase sectors in the valid secure user area (unless the application is executed from a valid secure user area);
- Erase the valid erase / program protection sectors;
- Erase the area not belong to the main flash block.

The following program operations will be ignored and the WPERR bit in the FMC_STAT register will be set:

- When SPC is protection level low, in debug mode, boot from SRAM or boot from bootloader mode, program the main flash block.
- Program sectors in the valid DCRP area unless executing GD secure library;
- Program sectors in the valid secure user area unless executing user security code or GD secure library;
- Program the valid erase/program protection sectors.
- Program the area not belong to the main flash block.

If WPERRIE bit in FMC_CTL register is set to 1, an interrupt is generated when WPERR flag is set. The software can clear it by writing 1.

### Program sequence error (PGSERR)

The following operation will set the PGSERR in the FMC_STAT register, and the current program operation is aborted:

- When a program operation is requested but the program enable bit (PG) has not been set in FMC_CTL register
- When PGCHEN bit is enable but the programming area is not all 0xFF before programming.

**Note:** If PGCHEN is set and burst program is enable, FMC will check the flash data according to the burst size. If it is checked that the data is not all 0xFF, this beat program is failed and PGSERR is set. But other beats can be programmed normally.

If PGSERRIE bit in FMC_CTL register is set to 1, an interrupt is generated when PGSERR flag is set. The software can clear it by writing 1.

### Read protection error (RPERR)

When attempt a read operation to a DCRP, a SPC protected area, FMC will set the RPERR in the FMC_STAT register, and the current read operation is aborted, but user can request a new read operation.

If RPERRIE bit in FMC_CTL register is set to 1, an interrupt is generated when RPERR flag is set. The software can clear it by writing 1.

### Read secure error (RSERR)

When attempt a read operation to a secure user area, FMC will set the RSERR in the FMC_STAT register, and the current read operation is aborted.

If RSERRIE bit in FMC_CTL register is set to 1, an interrupt is generated when RSERR flag is set. The software can clear it by writing 1.

### Option byte modify error (OBMERR)

When an error occurs during an option change operation, FMC will set the OBMERR flag in the FMC_STAT register, and the current operation is aborted.

If OBMERRIE bit in FMC_CTL register is set to 1, an interrupt is generated when OBMERR flag is set. The software can clear it by writing 1.

### Hard fault errors

The following operation will generate a bus error:

- When SPC protection level low , in debug mode, boot from SRAM or boot from boot

loader mode, access main flash block.

- Access secure user area without correct access rights.
- Access address out of range.
- Any wrong sequence to unlock the FMC_CTL or FMC_OBCTL register.

### 3.3.15. FMC interrupts

The FMC interrupt events and flags are listed in ***Table 3-9 FMC interrupt requests***.

**Table 3-9 FMC interrupt requests**

| Flag | Description | Clear method | Interrupt enable bit |
|---|---|---|---|
| ENDF | End of operation | Write 1 to corresponding bit in FMC_STAT register | ENDIE |
| WPERR | Write protection error | | WPERRIE |
| PGSERR | Program sequence error | | PGSERRIE |
| RPERR | Read protection error | | RPERRIE |
| RSERR | Read secure error | | RSERRIE |
| OBMERR | Option bytes modify error | | OBMERRIE |

## 3.4. Register definition

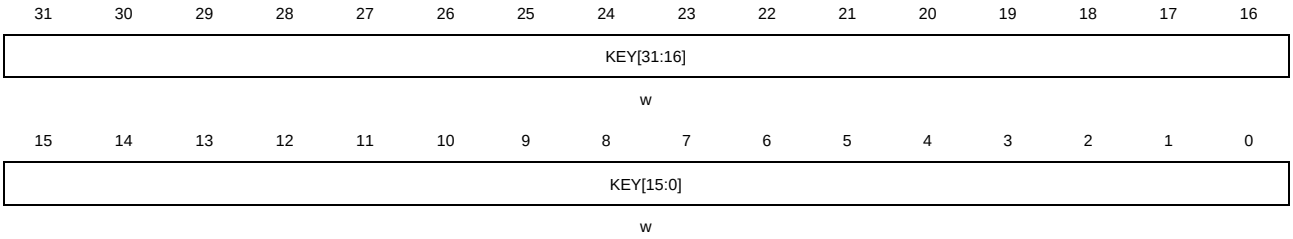
FMC base address: 0x5200 2000

### 3.4.1. Unlock key register (FMC_KEY)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



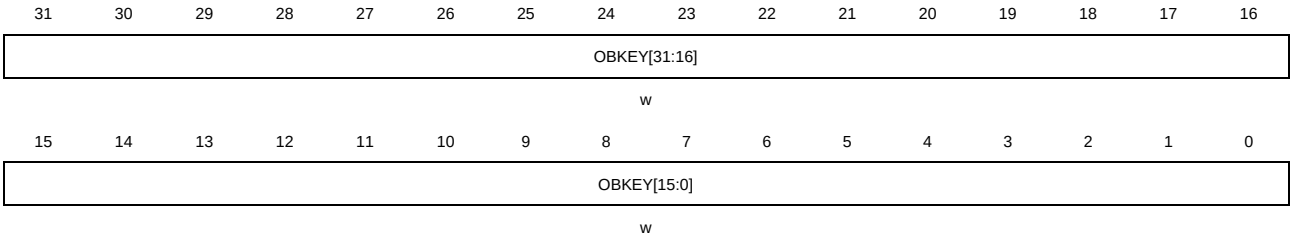

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | KEY[31:0] | FMC_CTL unlock register<br>These bits are only be written by software.<br>Write KEY [31:0] with keys to unlock FMC_CTL register. |

### 3.4.2. Option byte unlock key register (FMC_OBKEY)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | OBKEY[31:0] | These bits are only be written by software.<br>Write OBKEY [31:0] with keys to unlock FMC_OBCTL register. |

### 3.4.3. Control register (FMC_CTL)

Address offset: 0xC

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | RSERRIE | RPERRIE | Reserved | | | | PGSERRIE | WPERRIE | ENDIE |
| | | | | | | | rw | rw | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | START | Reserved | | PGCHEN | MER | SER | PG | LK |
| | | | | | | | | rs | | | rw | rw | rw | rw | rs |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | RSERRIE | Read secure error interrupt enable.<br>This bit is set or cleared by software only when LK is set to 0.<br>0: Disable read secure error interrupt<br>1: Enable read secure error interrupt |
| 23 | RPERRIE | Read protection error interrupt enable<br>This bit is set or cleared by software only when LK is set to 0<br>0: Disable read protection error interrupt<br>1: Enable read protection error interrupt |
| 22:19 | Reserved | Must be kept at reset value. |
| 18 | PGSERRIE | Program sequence error interrupt enable<br>This bit is set or cleared by software only when LK is set to 0<br>0: Disable program sequence error interrupt<br>1: Enable program sequence error interrupt |
| 17 | WPERRIE | Erase/program protection error interrupt enable<br>This bit is set or cleared by software only when LK is set to 0.<br>0: Disable erase/program error interrupt<br>1: Enable erase/program error interrupt |
| 16 | ENDIE | End of operation interrupt enable<br>This bit is set or cleared by software when LK is set to 0.<br>0: Disable end of operation interrupt<br>1: Enable end of operation interrupt |
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | START | Send erase command to FMC<br>This bit is set by software to send erase command to FMC when LK is set to 0.<br>This bit is cleared by hardware when the BUSY bit is cleared. |
| 6:5 | Reserved | Must be kept at reset value. |
| 4 | PGCHEN | Check programming area enable<br>This bit is set or cleared by software only when LK is set to 0<br>0: Disable check whether the programming area is all 0xFF before programming.<br>1: Enable check whether the programming area is all 0xFF before programming. |

| | | |
|---|---|---|
| | | If this bit is set, and the programming area is not all 0xFF, PGSERR flag is set. And the program operation is invalid. |
| 3 | MER | Mass erase command bit<br>This bit is set or cleared by software when LK is set to 0.<br>0: No effect<br>1: Main flash mass erase command<br>If MER and SER are both set, FMC will perform a mass erase operation. Because the priority of MER is higher than SER. |
| 2 | SER | Sector erase command bit<br>This bit is set or clear by software when LK is set to 0.<br>0: No effect<br>1: Main Flash sector erase command<br>If MER and SER are both set, FMC will perform a mass erase operation. Because the priority of MER is higher than SER. |
| 1 | PG | Main Flash program command bit<br>This bit is set or clear by software when LK is set to 0.<br>0: No effect<br>1: Main Flash program command |
| 0 | LK | FMC_CTL lock bit<br>This bit is cleared by hardware when right sequence written to the FMC_KEY register. This bit can be set by software. |

**Note:** This register should be reset after the corresponding Flash operation completed.

### 3.4.4. Status register (FMC_STAT)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | OBMERR | Reserved | | | | | RSERR | RPERR | Reserved | | | | PGSERR | WPERR | ENDF |
| | rc_w1 | | | | | | rc_w1 | rc_w1 | | | | | rc_w1 | rc_w1 | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | BUSY |
| | | | | | | | | | | | | | | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | OBMERR | Option byte modify error flag .<br>This bit is set by hardware. The software can clear it by writing 1.<br>0: Disable option byte modify error |

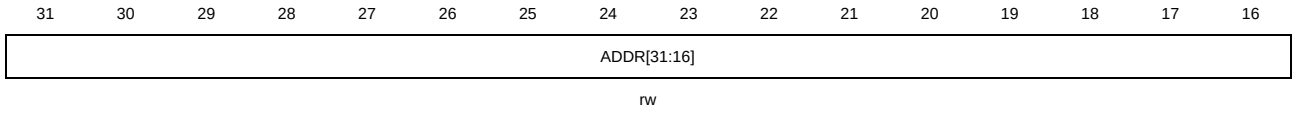
| | | |
|---|---|---|
| | | 1: Enable option byte modify error |
| 29:25 | Reserved | Must be kept at reset value. |
| 24 | RSERR | Read secure error flag bit.<br>When access a secure user protected address, this bit is set by hardware. The software can clear it by writing 1.<br>0: No read secure error occurs<br>1: Read secure error occurs |
| 23 | RPERR | Read protection error flag bit.<br>When access address protected by DCRP or RDP, this bit is set by hardware. The software can clear it by writing 1.<br>0: No read protection error occurs<br>1: Read protection error occurs |
| 22:19 | Reserved | Must be kept at reset value. |
| 18 | PGSERR | Program sequence error flag bit.<br>When a program sequence error occurs, this bit is set by hardware. The software can clear it by writing 1.<br>0: No program sequence error occurs<br>1: Program sequence error occurs |
| 17 | WPERR | Erase/program protection error flag bit.<br>When a erase/program protection error occurs, this bit is set by hardware. The software can clear it by writing 1.<br>0: No write protection error occurs<br>1: Write protection error occurs |
| 16 | ENDF | End of operation flag bit.<br>When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1. |
| 15:1 | Reserved | Must be kept at reset value. |
| 0 | BUSY | The Flash is busy bit<br>When the operation is in progress, this bit is set to 1. When the operation is end or an error is generated, this bit is cleared to 0. |

### 3.4.5. Address register (FMC_ADDR)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR[31:0] | Flash erase command address bits<br>These bits are configured by software.<br>ADDR bits are the address of flash to be erased. |

### 3.4.6. Option byte control register (FMC_OBCTL)

Address offset: 0x18

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | OBMERRIE | Reserved | | | | | | | | | | | | | |
| | rw | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | OBSTART | OBLK |
| | | | | | | | | | | | | | | rs | rs |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | OBMERRIE | Option byte modify error interrupt enable.<br>This bit is set or cleared by software only when OBLK is set to 0.<br>0: Disable option byte modify error interrupt<br>1: Enable option byte modify error interrupt |
| 29:2 | Reserved | Must be kept at reset value. |
| 1 | OBSTART | Send option byte change command to FMC bit.<br>This bit is set by software to send option byte change command to FMC only when OBLK is set to 0.<br>This bit is cleared by hardware when the BUSY bit is cleared |
| 0 | OBLK | FMC_OBCTL lock bit<br>This bit is cleared by hardware when right sequence written to FMC_OBKEY register.<br>This bit can be set by software. |

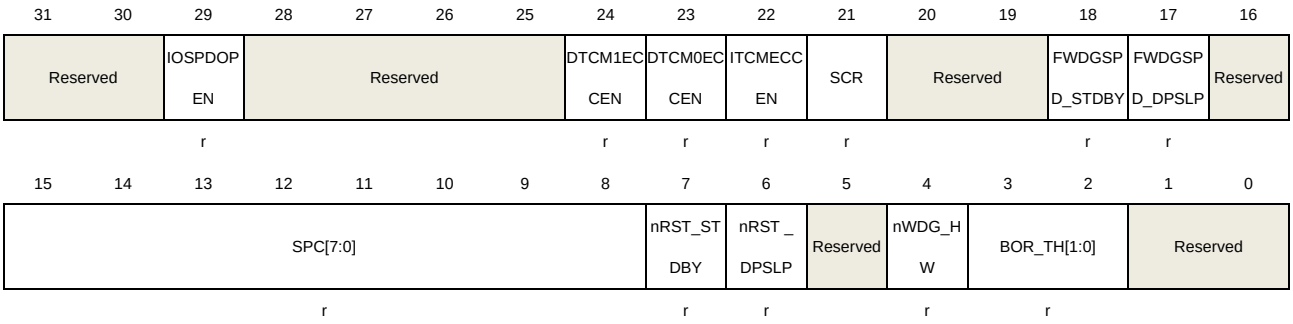
### 3.4.7. Option byte status register 0 (FMC_OBSTAT0_EFT)

Address offset: 0x1C

Reset value: 0xXXXX XXXX. Factory value is 0x01C6 AAD0

This register is the effective values of corresponding option bits. Load flash values after reset.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | IOSPDOPEN | Allowed enable status bit for I/O speed optimization at low-voltage.<br>0: Chip operating voltage greater than 2.5V, so I/O speed optimization is not allowed<br>1: Chip operating voltage is less than 2.5V, so I/O speed optimization is allowed |
| 28:25 | Reserved | Must be kept at reset value. |
| 24 | DTCM1ECCEN | DTCM1 ECC function enable status bit<br>0: Disable CPU DTCM1 ECC function<br>1: Enable CPU DTCM1 ECC function |
| 23 | DTCM0ECCEN | DTCM0 ECC function enable status bit<br>0: Disable CPU DTCM0 ECC function<br>1: Enable CPU DTCM0 ECC function |
| 22 | ITCMECCEN | ITCM ECC function enable status bit<br>0: Disable CPU ITCM ECC function<br>1: Enable CPU ITCM ECC function |
| 21 | SCR | Secure mode status bit<br>0: Disable secure mode.<br>1: Enable secure mode. |
| 20:19 | Reserved | Must be kept at reset value. |
| 18 | FWDGSPD_STDBY | FWDGT suspend option in standby mode status bit<br>0: FWDGT is suspend in system standby mode<br>1: FWDGT is running in system standby mode. |
| 17 | FWDGSPD_DPSLP | FWDGT suspend option in deepsleep mode status bit<br>0: FWDGT is suspend in system deepsleep mode<br>1: FWDGT is running in system deepsleep mode. |
| 16 | Reserved | Must be kept at reset value. |
| 15:8 | SPC[7:0] | Security protection level option byte status bits |

| | | |
|---|---|---|
| | | 0xAA: No protection<br>0xCC: Protection level high<br>Any value except 0xAA or 0xCC: Protection level low. |
| 7 | nRST_STDBY | Option byte standby reset status bit<br>0: Generates a reset instead of entering standby mode<br>1: No reset when entering standby mode. |
| 6 | nRST_DPSLP | Option byte deepsleep reset status bit<br>0: Generates a reset instead of entering Deep-sleep mode<br>1: No reset when entering Deep-sleep mode |
| 5 | Reserved | Must be kept at reset value. |
| 4 | nWDG_HW | Watchdog status bit<br>0: Hardware free watchdog<br>1: Software free watchdog |
| 3:2 | BOR_TH[1:0] | BOR threshold status bits<br>00: No BOR function<br>01: BOR threshold value 1<br>10: BOR threshold value 2<br>11: BOR threshold value 3 |
| 1:0 | Reserved | Must be kept at reset value. |

### 3.4.8. Option byte status register 0 (FMC_OBSTAT0_MDF)

Address offset: 0x20

Reset value: 0xXXXX XXXX

This register is used for modifying values to corresponding option bits. Values after reset is the effective values of the corresponding option bits.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | IOSPDOPEN | Reserved | | | | DTCM1ECCEN | DTCM0ECCEN | ITCMECCEN | SCR | Reserved | | FWDGSPD_STDBY | FWDGSPD_DPSLP | Reserved |
| | | rw | | | | | rw | rw | rw | rw | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPC[7:0] | | | | | | | | nRST_STDBY | nRST_DPSLP | Reserved | nWDG_HW | BOR_TH[1:0] | | Reserved | |
| rw | | | | | | | | rw | rw | | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | IOSPDOPEN | Allowed enable configuration bit for I/O speed optimization at low-voltage. |

| | | |
|---|---|---|
| | | 0: Chip operating voltage greater than 2.5V, so I/O speed optimization is not allowed<br>1: Chip operating voltage is less than 2.5V, so I/O speed optimization is allowed |
| 28:25 | Reserved | Must be kept at reset value. |
| 24 | DTCM1ECCEN | DTCM1 ECC function enable configuration bit<br>0: Disabled CPU DTCM1 ECC function<br>1: Enabled CPU DTCM1 ECC function |
| 23 | DTCM0ECCEN | DTCM0 ECC function enable configuration bit<br>0: Disabled CPU DTCM0 ECC function<br>1: enabled CPU DTCM0 ECC function |
| 22 | ITCMECCEN | ITCM ECC function enable configuration bit<br>0: disabled CPU ITCM ECC function<br>1: enabled CPU ITCM ECC function |
| 21 | SCR | Secure mode configuration bit<br>0: Disable secure mode.<br>1: Enable secure mode. |
| 20:19 | Reserved | Must be kept at reset value. |
| 18 | FWDGSPD_STDBY | FWDGT suspend option in standby mode configuration bit<br>0: FWDGT is suspend in system standby mode<br>1: FWDGT is running in system standby mode. |
| 17 | FWDGSPD_DPSLP | FWDGT suspend option in deepsleep mode configuration bit<br>0: FWDGT is suspend in system deepsleep mode<br>1: FWDGT is running in system deepsleep mode. |
| 16 | Reserved | Must be kept at reset value. |
| 15:8 | SPC[7:0] | Security protection level option configuration bits<br>0xAA: No protection<br>0xCC: Protection level high<br>Any value except 0xAA or 0xCC: Protection level low. |
| 7 | nRST_STDBY | Option byte standby reset configuration bit<br>0: Generates a reset instead of entering standby mode<br>1: No reset when entering standby mode |
| 6 | nRST_DPSLP | Option byte deepsleep reset configuration bit<br>0: Generates a reset instead of entering deep-sleep mode<br>1: No reset when entering deep-sleep mode |
| 5 | Reserved | Must be kept at reset value. |
| 4 | nWDG_HW | Watchdog configuration bit<br>0: Hardware free watchdog<br>1: Software free watchdog |

| | | |
|---|---|---|
| 3:2 | BOR_TH[1:0] | BOR threshold configuration bits<br>00: No BOR function<br>01: BOR threshold value 1<br>10: BOR threshold value 2<br>11: BOR threshold value 3 |
| 1:0 | Reserved | Must be kept at reset value. |

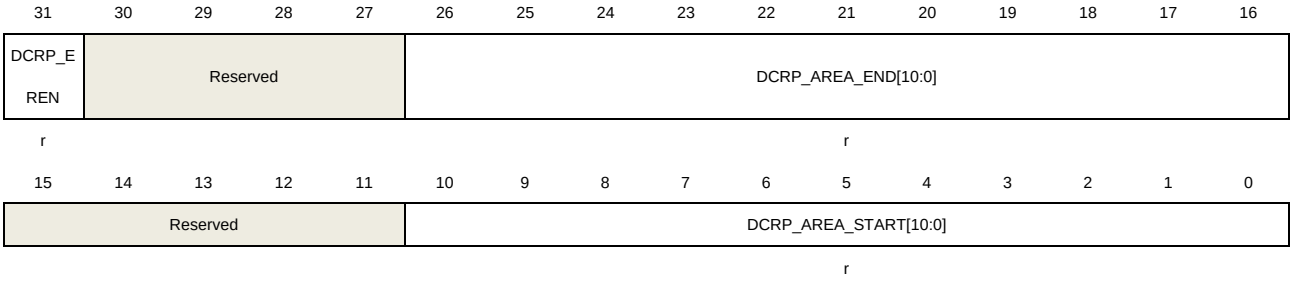
### 3.4.9. DCRP address register (FMC_DCRPADDR_EFT)

Address offset: 0x28

Reset value: 0xXXXX 0XXX. Factory value is 0x0000 00FF

This register is the effective values of corresponding option bits. Load flash values after reset

This register has to be accessed by word (32-bit).



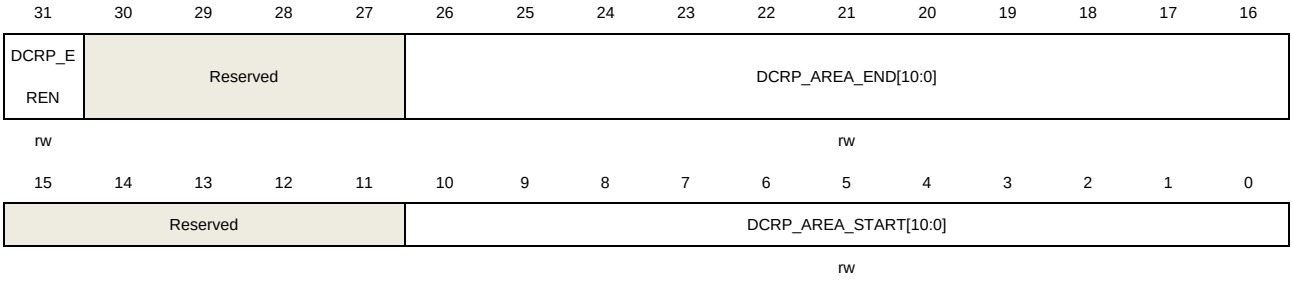

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DCRP_EREN | DCRP area erase enable status bit.<br>0: DCRP is not erased.<br>1: DCRP is erased when a SPC level low to no protection demotion or a protection-removed mass erase occur. |
| 30:27 | Reserved | Must be kept at reset value. |
| 26:16 | DCRP_AREA_END[10:0] | DCRP area end address status bits<br>These bits contain the last 4K-byte block of the DCRP area.<br>End absolute address = ( DCRP_AREA_END[10:0] + 1 ) * 4096 – 1 + 0x0800_0000.<br>If DCRP_AREA_END[10:0] = DCRP_AREA_START[10:0], DCRP protects the whole main flash block.<br>If DCRP_AREA_END[10:0] < DCRP_AREA_START[10:0], protection is invalid. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | DCRP_AREA_START[10:0] | DCRP area start address status bits<br>These bits contain the first 4K-byte block of the DCRP area.<br>Start absolute address = DCRP_AREA_START[10:0] * 4096 + 0x0800_0000.<br>If DCRP_AREA_END[10:0] = DCRP_AREA_START[10:0], DCRP protects the whole main flash block.<br>If DCRP_AREA_END[10:0] < DCRP_AREA_START[10:0], protection is invalid. |

### 3.4.10. DCRP address register (FMC_DCRPADDR_MDF)

Address offset: 0x2C
Reset value: 0xXXXX 0XXX

This register is used for modifying values to corresponding option bits. Values after reset is the effective values of the corresponding option bits.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DCRP_EREN | DCRP area erase enable configuration bit.<br>0: DCRP is not erased.<br>1: DCRP is erased when a SPC level low to no protection demotion or a protection-removed mass erase. |
| 30:27 | Reserved | Must be kept at reset value. |
| 26:16 | DCRP_AREA_END[10:0] | DCRP area end address configuration bits<br>These bits contain the last 4K-byte block of the DCRP area.<br>End absolute address = ( DCRP_AREA_END[10:0] + 1 ) * 4096 – 1 + 0x0800_0000<br>If DCRP_AREA_END[10:0] = DCRP_AREA_START[10:0], DCRP protects the whole main flash block.<br>If DCRP_AREA_END[10:0] < DCRP_AREA_START[10:0], protection is invalid. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | DCRP_AREA_START[10:0] | DCRP area start address configuration bits<br>These bits contain the first 4K-byte block of the DCRP area.<br>Start absolute address = DCRP_AREA_START[10:0] * 4096 + 0x0800_0000.<br>If DCRP_AREA_END[10:0] = DCRP_AREA_START[10:0], DCRP protects the whole main flash block.<br>If DCRP_AREA_END[10:0] < DCRP_AREA_START[10:0], protection is invalid. |

### 3.4.11. Secure address register (FMC_SCRADDR_EFT)

Address offset: 0x30
Reset value: 0xXXXX 0XXX. Factory value is 0x0000 00FF

This register is the effective values of corresponding option bits. Load flash values after reset

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_EREN | Reserved | | | | SCR_AREA_END[10:0] | | | | | | | | | | |
| r | | | | | r | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | SCR_AREA_START[10:0] | | | | | | | | | | |
| | | | | | r | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | SCR_EREN | Secure user area erase enable option status bit.<br>0: No action<br>1: The secure user area is erased when a SPC level low to no protection demotion or a protection-removed mass erase. |
| 30:27 | Reserved | Must be kept at reset value. |
| 26:16 | SCR_AREA_END[10:0] | Secure user area end address status bits<br>These bits contain the last 4K-byte block of the secure user area.<br>End absolute address = (SCR_AREA_END[10:0] + 1 ) * 4096 – 1 + 0x0800_0000<br>If SCR_AREA_END[10:0] = SCR_AREA_START[10:0], whole main flash block is secure user area.<br>If SCR_AREA_END[10:0] < SCR_AREA_START[10:0], protection is invalid. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | SCR_AREA_START[10:0] | Secure user area start address status bits<br>These bits contain the first 4K-byte block of the secure user area.<br>Start absolute address = SCR_AREA_START[10:0] * 4096 + 0x0800_0000.<br>If SCR_AREA_END[10:0] = SCR_AREA_START[10:0], whole main flash block is secure user area.<br>If SCR_AREA_END[10:0] < SCR_AREA_START[10:0], protection is invalid. |

### 3.4.12. Secure address register (FMC_SCRADDR_MDF)

Address offset: 0x34
Reset value: 0xXXXX 0XXX

This register is used for modifying values to corresponding option bits. Values after reset is the effective values of the corresponding option bits.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_EREN | Reserved | | | | SCR_AREA_END[10:0] | | | | | | | | | | |
| rw | | | | | r | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | SCR_AREA_START[10:0] |
|---|---|
| | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | SCR_EREN | Secure user area erase enable option configuration bit.<br>0: No action<br>1: The secure user area is erased when a SPC level low to no protection demotion or a protection-removed mass erase. |
| 30:27 | Reserved | Must be kept at reset value. |
| 26:16 | SCR_AREA_END[10:0] | Secure user area end address configuration bits<br>These bits contain the last 4K-byte block of the secure user area.<br>End absolute address = (SCR_AREA_END[10:0] + 1 ) * 4096 – 1 + 0x0800_0000<br>If SCR_AREA_END[10:0] = SCR_AREA_START[10:0], whole main flash block is secure user.<br>If SCR_AREA_END[10:0] < SCR_AREA_START[10:0], protection is invalid. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | SCR_AREA_START[10:0] | Secure user area start address configuration bits<br>These bits contain the first 4K-byte block of the secure user area.<br>Start absolute address = SCR_AREA_START[10:0] * 4096 + 0x0800_0000.<br>If SCR_AREA_END[10:0] = SCR_AREA_START[10:0], whole main flash block is secure user.<br>If SCR_AREA_END[10:0] < SCR_AREA_START[10:0], protection is invalid. |

## 3.4.13. Erase/program protection register (FMC_WP_EFT)

Address offset: 0x38

Reset value: 0xXXXX XXXX .Factory value is 0x3FFF FFFF

This register is the effective values of corresponding option bits. Load flash values after reset

This register is read-only. This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | WP[21:16] | | | | | |
| | | | | | | | | | | r | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:0 | WP[21:0] | Sector erase/program protection option status bit<br>In WP[21], each bit reflects the corresponding 64 sectors to erase/program |

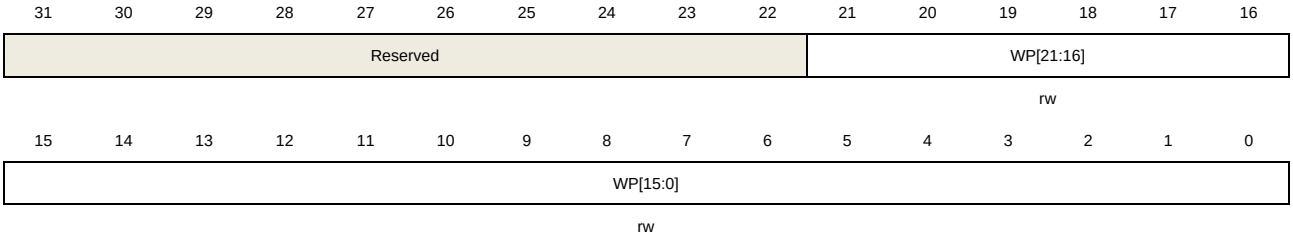
protection status.
0: corresponding 64 sectors are erase/program protected.
1: corresponding 64 sectors are not erase/program protected.
In WP[20:16], each bit reflects the corresponding 128 sectors to erase/program protection status.
0: corresponding 128 sectors are erase/program protected.
1: corresponding 128 sectors are not erase/program protected.
In WP[15:0], each bit reflects the corresponding 16 sector to erase/program protection status.
0: corresponding 16 sector is erase/program protected.
1: corresponding 16 sector is not erase/program protected.

### 3.4.14. Erase/program protection register (FMC_WP_MDF)

Address offset: 0x3C
Reset value: 0xXXXX XXXX

This register is used for modifying values to corresponding option bits. Values after reset is the effective values of the corresponding option bits.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:0 | WP | Sector erase/program protection option configuration bit<br>In WP[21], each bit can set the corresponding 64 sectors to erase/program protection status.<br>0: set corresponding 64 sectors to erase/program protected.<br>1: set corresponding 64 sectors to not erase/program protected.<br>In WP[20:16], each bit can set the corresponding 128 sectors to erase/program protection status.<br>0: set corresponding 128 sectors to erase/program protected.<br>1: set corresponding 128 sectors to not erase/program protected.<br>In WP[15:0], each bit can set the corresponding 16 sector to erase/program protection status.<br>0: set corresponding 16 sector to erase/program protected.<br>1: set corresponding 16 sector to not erase/program protected. |

### 3.4.15. Boot address register (FMC_BTADDR_EFT)

Address offset: 0x40

Reset value: 0xXXXX XXXX. Factory value is 0x1FF0 0800

This register is the effective values of corresponding option bits. Load flash values after reset

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOOT_ADDR1[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BOOT_ADDR0[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | BOOT_ADDR1[15:0] | Boot address 1 status bits<br>MSB of the boot address if BOOT pin is high. |
| 15:0 | BOOT_ADDR0[15:0] | Boot address 0 status bits<br>MSB of the boot address if BOOT pin is low. |

### 3.4.16. Boot address register (FMC_BTADDR_MDF)

Address offset: 0x44

Reset value: 0xXXXX XXXX

This register is used for modifying values to corresponding option bits. Values after reset is the effective values of the corresponding option bits.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOOT_ADDR1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BOOT_ADDR0[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | BOOT_ADDR1[15:0] | Boot address 1 configuration bits.<br>Configure the MSB of boot address if the BOOT pin is high. |
| 15:0 | BOOT_ADDR0[15:0] | Boot address 0 configuration bits.<br>Configure the MSB of the boot address if the BOOT pin is low. |

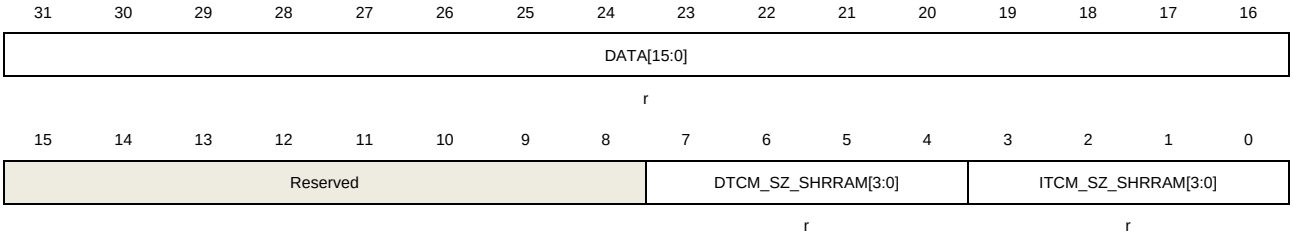
### 3.4.17. Option byte status register 1 (FMC_OBSTAT1_EFT)

Address offset: 0x50

Reset value: 0xXXXX 0XXX. Factory value is 0x0000 0087

This register is the effective values of corresponding option bits. Load flash values after reset

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | DATA[15:0] | User defined option byte data status value |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:4 | DTCM_SZ_SHRRAM[3:0] | DTCM size of shared RAM status bits.<br>DTCM + ITCM can not be more than 512Kbyte<br>0000: 0-byte DTCM<br>0001~0110: Reserved<br>0111: 64-Kbyte DTCM<br>1000: 128-Kbyte DTCM<br>1001: 256-Kbyte DTCM<br>1010: 512-Kbyte DTCM<br>1011~1111: Reserved |
| 3:0 | ITCM_SZ_SHRRAM[3:0] | ITCM size of shared RAM status bits.<br>DTCM + ITCM can not be more than 512Kbyte<br>0000: 0-byte ITCM<br>0001~0110: Reserved<br>0111: 64-Kbyte ITCM<br>1000: 128-Kbyte ITCM<br>1001: 256-Kbyte ITCM<br>1010: 512-Kbyte ITCM<br>1011~1111: Reserved |

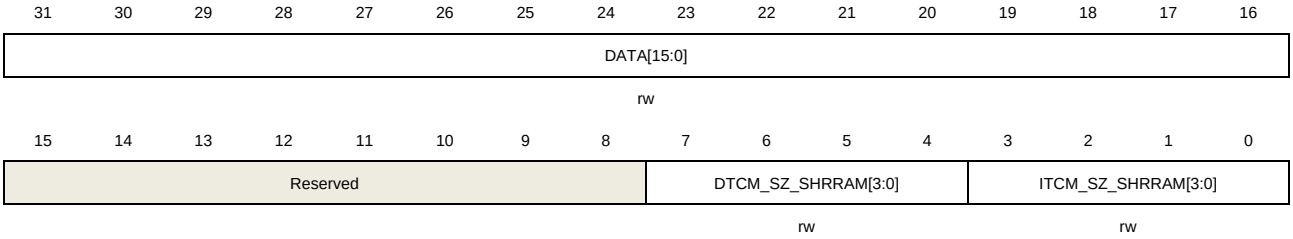
### 3.4.18. Option byte status register 1 (FMC_OBSTAT1_MDF)

Address offset: 0x54

Reset value: 0xXXXX 0XXX.

This register is used for modifying values to corresponding option bits. Values after reset is the effective values of the corresponding option bits.

This register has to be accessed by word (32-bit).



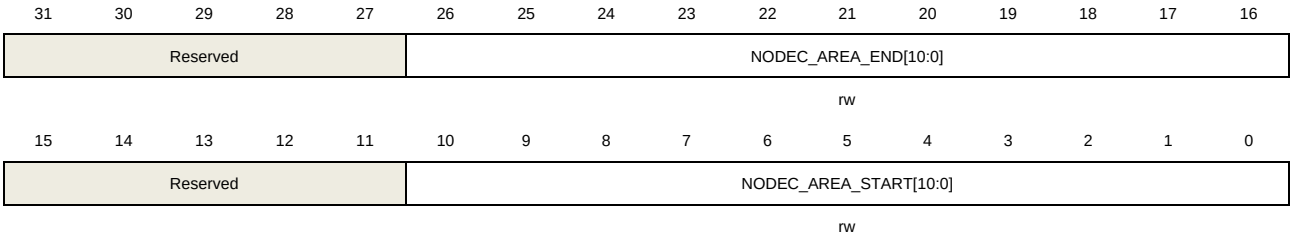

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | DATA[15:0] | User defined option byte data configuration value |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:4 | DTCM_SZ_SHRRAM[3:0] | DTCM size of shared RAM configuration bits.<br>DTCM + ITCM can not be more than 512Kbyte<br>0000: 0-byte DTCM<br>0001~0110: Reserved<br>0111: 64-Kbyte DTCM<br>1000: 128-Kbyte DTCM<br>1001: 256-Kbyte DTCM<br>1010: 512-Kbyte DTCM<br>1011~1111: Reserved |
| 3:0 | ITCM_SZ_SHRRAM[3:0] | ITCM size of shared RAM configuration bits.<br>DTCM + ITCM can not be more than 512Kbyte<br>0000: 0-byte ITCM<br>0001~0110: Reserved<br>0111: 64-Kbyte ITCM<br>1000: 128-Kbyte ITCM<br>1001: 256-Kbyte ITCM<br>1010: 512-Kbyte ITCM<br>1011~1111: Reserved |

## 3.4.19. NO-RTDEC area register (FMC_NODEC)

Address offset: 0x60

Reset value: 0x0000 00FF

This register has to be accessed by word (32-bit) only when LK is set to 0.



| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26:16 | NODEC_AREA_END[10:0] | NO-RTDEC area end address<br>These bits contain the last 4K-byte block that reading main flash block without decryption.<br>End absolute address= NODEC_AREA_END[10:0] * 4096 -1 + 0x0800_0000.<br>If NODEC_AREA_END[10:0] = NODEC_AREA_START[10:0], whole main flash block is reading without decryption.<br>If NODEC_AREA_END[10:0] < NODEC_AREA_START[10:0], whole main flash block is reading with decryption. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | NODEC_AREA_START[10:0] | No-RTDEC area start address<br>These bits contain the first 4K-byte block that reading main flash block without decryption.<br>Start absolute address = NODEC_AREA_START[10:0] * 4096 + 0x0800_0000.<br>If NODEC_AREA_END[10:0] = NODEC_AREA_START[10:0], whole main flash block is reading without decryption.<br>If NODEC_AREA_END[10:0] < NODEC_AREA_START[10:0], whole main flash block is reading with decryption. |

### 3.4.20. AES IV register (FMC_AESIVx_EFT) (x = 0…2)

Address offset: 0x68 + 0x4 * x
Reset value: 0xXXXX XXXX

This register is the effective high 96 bits of AES IV. The AES initialization vector is not an option byte, and it is stored in non-volatile AES IV area. The register value is loaded from this area after reset

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AESIV[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AESIV[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | AESIV[31:0] | AES initialization vector status value<br>The initialization vector AES_IV[127:0] = AESIV[95:0] \|\| 12'b0 \|\| ReadAddress[23:4].<br>The 96 bits AESIV[95:0] is formed with [AESIV2, AESIV1, AESIV0]. |

### 3.4.21. AES IV register (FMC_AESIVx_MDF) (x = 0…2)

Address offset: 0x74 + 0x4 * x
Reset value: 0x0000 0000

This register is used for modifying high 96 bits of AES IV.

This register has to be accessed by word (32-bit) only when LK is set to 0.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AESIV[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AESIV[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | AESIV[31:0] | AES initialization vector configuration value<br>The initialization vector AES_IV[127:0] = AESIV[95:0] \|\| 12'b0 \|\| ReadAddress[23:4].<br>The AESIV[95:0] is formed with [AESIV2, AESIV1,AESIV0].<br>After the IV is written to the FMC_AESIV2_MDF register, the values in the FMC_AESIV0/1/2_MDF registers will be updated to the AES IV area, and the BUSY bit will be automatically set to 1. When the update is complete, the BUSY bits automatically clear to 0.<br>**Note:** Before writing to FMC_AESIV2_MDF, user ensure that no program, erase, or option byte modification operations are inprogress. Otherwise, the FMC_AESIV2_MDF register cannot be written, and the update operation will not proceed. |

### 3.4.22. Product ID register x (FMC_PIDx) (x = 0,1)

Address offset: 0x100 + 0x4 * x
Reset value: 0xXXXX XXXX

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PID[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PID[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PID[31:0] | Product reserved ID code register x<br>These bits are read only by software.<br>These bits are unchanged constant after power on. These bits are one time program |

when the chip produced.

# 4. Electronic fuse (EFUSE)

## 4.1. Overview

The efuse controller has efuse macro that store system parameters. As a non-volatile unit of storage, the bit of efuse macro cannot be restored to 0 once it is programmed to 1.

## 4.2. Characteristics

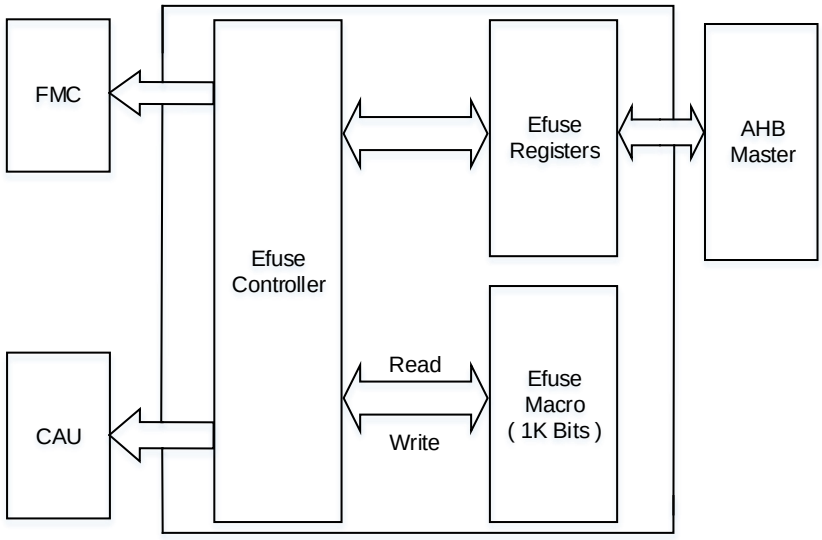
- One-time programmable nonvolatile efuse storage cells organized as 32*32 bits.
- Double-bit redundant backup mechanism.
- All bits in the efuse cannot be rollback from 1 to 0.
- Can only be accessed through corresponding register.
- Each bit in efuse macro can only be programmed once, and software must avoid reprogramming.
- Voltage range for program: 1.71~1.98 V.
- Voltage range for read: 0.72~1.05 V

## 4.3. Function overview

### 4.3.1. Block diagram

Efuse controller implements the efuse macro read-program control logic. The efuse macro contains a storage unit of 1K bits cells.

**Figure 4-1. Block diagram of efuse controller**



Efuse adopts the double-bit redundant backup mechanism. The first 512 bits data and the last 512 bits data are backed up with each other to effectively ensure the correctness of the data.When programming the Nth bit of the efuse, the efuse controller will program both the Nth and (N+512)th bit. When reading the Nth bit of efuse, the fuse controller will read both the Nth and (N+512)th bit, and perform ORed operation on the read results of 2 bits, and finally return the operation result to the corresponding parameter register. All the above processes are internally processed by the chip. The user only needs to access the first 512 bits, and the last 512 bits of data cannot be accessed by the user.

### 4.3.2. Efuse macro description

The efuse macro stores 5 system parameters, every system parameters have different width. Each system parameter in efuse has its own read and program protection attributes.

Program protection attribute is list below:

- User control parameter:
  The parameter stored in efuse macro can be modified multiple times, but once the bit has been programmed to 1, hardware will prohibit the bit being programmed again. Only when SCRLK bit is 0, high 16 bits of the register can be modified by user, and only when UCLK bit is 0, low 16 bits of the register can be modified by user. But the modified value of register will not be stored in efuse macro, unless a efuse ***Program operation*** is executed successfully. After the efuse program operation, user need a power reset ( for JTAGNSW and NDBG[1:0] ) or a system reset ( bits except JTAGNSW and NDBG[1:0] ) to load the parameter from efuse macro into the register but this newly modified parameter takes effect only after the power reset ( for JTAGNSW and NDBG[1:0] ) or a system reset ( bits except JTAGNSW and NDBG[1:0] ).

- MCU reserved parameter:
  The parameter stored in efuse macro can be modified multiple times, but once the bit has been programmed to 1, software must prohibit the bit being programmed again in user application ( by setting the already programmed bit in register to 0 when efuse program operation ). The register can be modified by user, but the modified value of register will not be stored in efuse macro, unless a efuse ***Program operation*** is executed successfully. After the efuse program operation, user need a system reset or a efuse ***Read operation*** to load the parameter from efuse macro into the register, but the DCRP_AREA_END[7:0]、DCRP_AREA_START[7:0]、DCRPLK、AESNCAU (AESNCAU only for GD32H737_757_759) bits takes effect only after the system is reset.

- Debug password parameter:
  The parameter stored in efuse macro can be modified multiple times, but once the bit has been programmed to 1, software must prohibit the bit being programmed again in user application ( by setting the already programmed bit in register to 0 when efuse program operation ). Only when DPLK bit is 0 the register can be modified by user, but the modified value of register will not be stored in efuse macro, unless a efuse ***Program operation*** is executed successfully. After the efuse program operation, user need a

system reset or a efuse ***Read operation*** to load the parameter from efuse macro into the register and this newly modified Debug password parameter takes effect only after the system is reset.

- AES key parameter:
  The parameter stored in efuse macro can be modified multiple times, but once the bit has been programmed to 1, software must prohibit the bit being programmed again in user application ( by setting the already programmed bit in register to 0 when efuse program operation ). And only when AESEN bit is 0, the register can be modified by user, but the modified value of register will not be stored in efuse macro, unless a efuse ***Program operation*** is executed successfully. After the efuse program operation, user need a system reset becuse this newly modified parameter takes effect only after the system is reset.

- User data parameter:
  The parameter stored in efuse macro can be modified multiple times, but once the bit has been programmed to 1, software must prohibit the bit being programmed again in user application ( by setting the already programmed bit in register to 0 when efuse program operation ). And only when UDLK bit is 0, the register can be modified by user, but the modified value of register will not be stored in efuse macro, unless a efuse ***Program operation*** is executed successfully. After the efuse program operation, user need a system reset or a efuse ***Read operation*** to load the parameter from efuse macro into the register.

Read protection attribute is list below:

- User control parameter:
  The register can be read without restriction. After system reset, except JTAGNSW, NDBG[1:0] bits, all other bits in EFUSE_USER_CTL register will be restored to the value of parameter which is read out from efuse. After power reset, JTAGNSW, NDBG[1:0] will be restored to the value of parameter which is read out from efuse.

- MCU reserved parameter:
  The register can be read. After system reset, the register will be restored to the value of parameter which is read out from efuse. User can also read out the data from efuse macro by configuring the control register to perform the ***Read operation***.

- Debug password parameter:
  When the DPLK bit is 1, the JTAGNSW bit is 1, and the NDBG[1:0] bits are 2b'01 or 2b'11, the register can not be read. Otherwise this register can be read. After system reset, the register will be restored to the value of parameter which is read out from efuse. User can also read out the data from efuse macro by configuring the control register to perform the ***Read operation***.

- AES key parameter:
  The register can not be read. But user can verify the correctness of the written AES key by AES key CRC function.
  **Note:** User must continuously write the complete 16 bytes AES key into the

EFUSE_AES_KEYx register to ensure that the CRC function can check the contents of all AES key.

- User data parameter
  The register can be read without restriction. After system reset, the register will be restored to the value of parameter which is read out from efuse. User can also read out the data from efuse macro by configuring the control register to perform the ***Read operation***.

The following table ***Table 4-1. System parameters*** shows the details of each efuse byte.

**Table 4-1. System parameters**

| Parameter | Width/ bytes | Start address | Program-protected | Read-protected | Description |
|---|---|---|---|---|---|
| User control | 4B | 10'd0 | The parameter in efuse macro can write multiple times, but cannot rollback. Hardware will prohibit the programmed bits being reprogrammed. | After system reset, corresponding bits in EFUSE_USER_CTL register will be restored to the value of parameter which is read out from efuse. | All other bits except JTAGNSW bit and NDBG[1:0] bits in ***User control register (EFUSE_USER_CTL)***. |
| | | | | After power reset, corresponding bits in EFUSE_USER_CTL register will be restored to the value of parameter which is read out from efuse. | JTAGNSW bit and NDBG[1:0] bits in ***User control register (EFUSE_USER_CTL)***. |
| MCU reserved | 4B | 10'd32 | The parameter in efuse macro can write multiple times, but cannot rollback. Software must prohibit the programmed bits being reprogrammed. | After system reset, the register will be restored to the value of parameter which is read out from efuse. And user can also read out the data from efuse macro by configuring the control register to perform the read operation. | MCU reserved parameter. For more details, refer to ***MCU reserved register (EFUSE_MCU_RSV)***. |
| Debug password | 8B | 10'd64 | | | When JTAGNSW=1, and NDBG[1:0] = 2b'01 or 2b'11, this parameter is used as the debug password for debug services. Otherwise, this parameter will be used as user data. For more details, refer to ***Debug password register x (EFUSE_DPx) (x = 0,1)*** |
| AES key | 16B | 10'd128 | | The parameter cannot be read out by user. But user can verify the correctness of the written AES key by AES key CRC | The AES key used to encrypt the firmware image. For more details, refer to ***Firmware AES key register x (EFUSE_AES_KE*** |

| Parameter | Width/ bytes | Start address | Program-protected | Read-protected | Description |
|---|---|---|---|---|---|
| | | | | function. | Yx) (x = 0...3). |
| User data | 16B | 10'd256 | | After system reset, the register will be restored to the value of parameter which is read out from efuse<br>The parameter stored in efuse can also be load into register by configuring the control register to perform the read operation. | User defined data.<br>For more details, refer to User data register x (EFUSE_USER_DATAx) (x = 0...3). |

**Note:** All parameters are custom parameters. The bits 10'd384~10'd511 of the fuse are used by the system and are not accessible by the user.

EFADDR[9:0] should be configured as the start address of the system parameter, and EFSIZE[4:0] should be configured as the width of the system parameter.

Read operation: One system parameter can be read at a time. It is forbidden to read multiple system parameters at the same time, otherwise an illegal access error will occur. However, if user only read a part of a system parameter, that is, a system parameter is not completely read, an illegal access error will not occur, but it may cause the readout data of the system parameter ar incorrect. Users should avoid this situation.

Program operation: The range of program operation cannot exceed the address range of a single system parameter, and multiple system parameters cannot be written at the same time, otherwise an illegal access error will occur.

### 4.3.3. Read operation

The value of the efuse can only be accessed through the corresponding register.

The following steps show the register access sequence of the efuse reading operation.

1. Clear the RDIF bit if it is set, and make sure there are no illegal access errors.
2. Reset the EFRW bit in EFUSE_CTL.
3. Write the desired efuse address and size to the EFUSE_ADDR register.
4. Set the EFSTR bit in EFUSE_CTL register.
5. Wait until the reading operations have been finished by checking the RDIF bit in EFUSE_STAT register.
6. Read the register value corresponding to the efuse.

When the read operation is executed successfully, the RDIF in EFUSE_STAT register is set, and an interrupt will be triggered if the RDIE bit in the EFUSE_CTL register is set.

**Note:** Efuse is very sensitive to current surges which will affect the result of read operation. It is strictly prohibited to perform read operation during the sequence of power-up or power-down, otherwise it will cause unpredictable consequences.

### 4.3.4. Program operation

The value of the efuse can only be modified through the corresponding register.

The following steps show the register access sequence of the efuse writing operation.

1. Clear the PGIF bit if it is set, and make sure there are no illegal access errors.
2. Set the EFRW bit in EFUSE_CTL.
3. Write the desired efuse address and size to the EFUSE_ADDR register.
4. Write the data to the corresponding register.
5. Set the EFSTR bit EFUSE_CTL register.
6. Wait until the writing operations have been finished by checking the PGIF bit in EFUSE_STAT register.

**Note:** If the data in the corresponding parameter register is all 0, after EFSTR is set to 1, the efuse macro will not be programmed, and PGIF will be automatically set to 1.

When the program operation is executed successfully, the PGIF in EFUSE_CTL register is set, and an interrupt will be triggered by efuse if the PGIE bit in the EFUSE_CTL register is set. It should be noted that the address and size of the written data must match the corresponding fuse register. If not match, IAERRIF bit in the EFUSE_STAT register will be set, and an interrupt will be triggered by efuse if the IAERRIE bit is set.

**Note:** Efuse is very sensitive to current surges which will affect the result of program operation. It is strictly prohibited to perform program operation during the sequence of power-up or power-down, otherwise it will cause unpredictable consequences.

### 4.3.5. AES key CRC function

The standard CRC-8-CCITT algorithm is used in CRC calculation. The CRC algorithm in this module is used to verify the values in the EFUSE_AES_KEYx registers or the AES key value stored in the efuse macro.

After continuously write 16 bytes AES key to the EFUSE_AES_KEYx registers whose offset address are 0x24, 0x28, 0x2C and 0x30, the hardware CRC module will automatically calculate the corresponding CRC checking code based on the AES key value in the EFUSE_AES_KEYx registers and store the calculation result to the AES_KEY_CRC bit field in the EFUSE_CTL register. At this time, user can compare the CRC checking code calculated by hardware with the software CRC checking code calculated by the user software. If the checking codes calculated by the software and hardware are the same, it indicates the 16 bytes AES key written to the EFUSE_AES_KEYx register is correct, otherwise the AES key written to the register is wrong.

When the AES key stored in efuse macro is read out after system reset, the hardware CRC module will automatically calculate the corresponding AES key CRC checking code based on the AES key value stored in efuse macro and store the calculation result to the AES_KEY_CRC bit field in the EFUSE_CTL register. At this time, user can compare the CRC checking code calculated by hardware with the software CRC checking code calculated by the user software. If the checking codes calculated by the software and hardware are the same, It indicates that the 16 bytes AES key that the user expects to write has been successfully and correctly written into efuse macro, otherwise the key written into efuse is wrong.

**Note:** CRC calculation result is generated after writing the EFUSE_AES_KEY3 register (offset address 0x30) or after the completion of the system reset read efuse.

### 4.3.6. EFUSE interrupts

The following operations will set the IAERRIF in the EFUSE_STAT register:

- The address is overstep boundry when reading and writing parameters in the efuse;
- When UCLK bit is set and effective, write the lower 16 bits of the EFUSE_USER_CTL;
- When SCRLK bit is set and effective, write the high 16 bits of the EFUSE_USER_CTL;
- When MCURSVLK bit is set and effective, write the lower 16 bits of the EFUSE_MCU_RSV;
- When DCRPLK bit is set and effective, write the high 16 bits of the EFUSE_MCU_RSV;
- When DPLK bit is set and effective, write the EFUSE_DPx;
- When DPLK bit is 1, JTAGNSW bit is 1, and NDBG[1:0] bits are 2b'01 or 2b'11, read the EFUSE_DPx;
- When AESEN bit is set and effective, write the EFUSE_AES_KEYx;
- When UDLK bit is set and effective, write the EFUSE_USER_DATAx;
- Read the User control parameter in efuse macro by the efuse read operation;
- Read the AES key parameter in efuse macro by the efuse read operation;

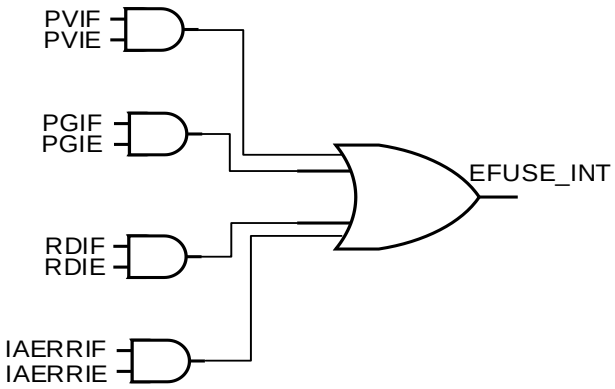
The efuse interrupt events and flags are listed in ***Table 4-2. EFUSE interrupt requests***.

**Table 4-2. EFUSE interrupt requests**

| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| Program voltage setting error interrupt | PVIF | PVIE |
| Program complete interrupt | PGIF | PGIE |
| Read complete interrupt | RDIF | RDIE |
| Illegal access error interrupt | IAERRIF | IAERRIE |

All of the interrupt events are ORed together before being sent to the interrupt controller, so the efuse can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine

**Figure 4-2 EFUSE interrupt mapping diagram**


PVIF
PVIE
PGIF
PGIE
RDIF
RDIE
IAERRIF
IAERRIE
EFUSE_INT

## 4.4. Register definition

EFUSE base address: 0x4002 2800

### 4.4.1. Control register (EFUSE_CTL)

Address offset: 0x00

Reset value: 0x7E00 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AES_KEY_CRC | | | | | | | | Reserved | | | | PVIE | RDIE | PGIE | IAERRIE |
| r | | | | | | | | | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MPVEN | Reserved | | | | | | | | | | | | | EFRW | EFSTR |
| rw | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | AES_KEY_CRC | 8-bits CRC calculation result value of AES key bits<br>This bit filed is used to verify the value of the EFUSE_AES_KEYx register and the AES key value stored in the efuse macro.<br>The standard CRC-8-CCITT algorithm is used in CRC calculation.<br>If AESEN is 0, there are two situations that will calculate the CRC value of AES key and store the CRC calculation result into this bit field:<br>(1) Continuously write 16 bytes AES key to the EFUSE_AES_KEYx registers whose offset address are 0x24, 0x28, 0x2C and 0x30. CRC calculation result is generated after writing the EFUSE_AES_KEY3 register (offset address 0x30).<br>(2) After system reset, all the AES key value is read out automatically from efuse macro by MCU. CRC calculation result is generated after the completion of the system reset read efuse. |
| 23:20 | Reserved | Must be kept at reset value. |
| 19 | PVIE | Program voltage setting error interrupt enable<br>0: Disable program voltage setting error interrupt<br>1: Enable program voltage setting error interrupt |
| 18 | RDIE | Read completed interrupt enable<br>0: Disable read completed interrupt<br>1: Enable read completed interrupt |
| 17 | PGIE | Program completed interrupt enable<br>0: Disable program completed interrupt<br>1: Enable program completed interrupt |
| 16 | IAERRIE | Illegal access error interrupt enable<br>0: Disable the illegal access error interrupt |

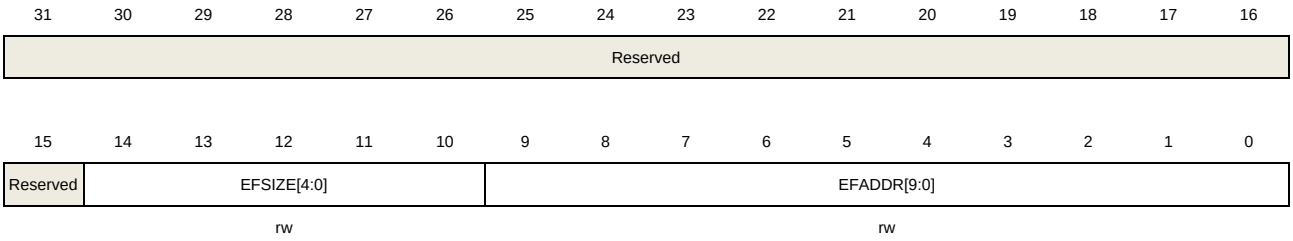
| | | |
|---|---|---|
| | | 1: Enable the illegal access error interrupt<br>This bit cannot be modified when the EFSTR bit in EFUSE_CTL register is 1 |
| 15 | MPVEN | Monitor program voltage function enable<br>0: Disable monitor program voltage function<br>1: Enable monitor program voltage function<br>These bits cannot be modified when the EFSTR bit in EFUSE_CTL register is 1. |
| 14:2 | Reserved | Must be kept at reset value. |
| 1 | EFRW | The selection of efuse operation<br>0: Read efuse<br>1: Write efuse<br>This bit cannot be modified when the EFSTR bit in EFUSE_CTL register is 1 |
| 0 | EFSTR | Start efuse operation<br>This bit is set by software and cleared by hardware<br>0: No effect<br>1: Start read or write efuse operation |

### 4.4.2. Address register (EFUSE_ADDR)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



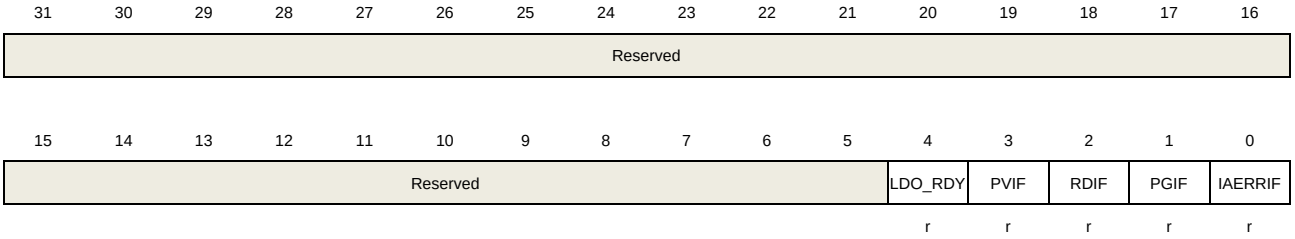

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14:10 | EFSIZE[4:0] | Read or write efuse data size<br>The size is calculated by byte.<br>These bits cannot be modified when the EFSTR bit in EFUSE_CTL register is 1. |
| 9:0 | EFADDR[9:0] | Read or write efuse data start address<br>EFADDR[9] must be set to 0, because the user cannot access data with an address greater than 512 bits, otherwise IAERRIF bit in EFUSE_STAT register is set.<br>These bits cannot be modified when the EFSTR bit in EFUSE_CTL register is 1. |

### 4.4.3. Status register (EFUSE_STAT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



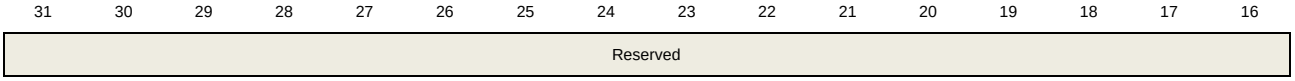

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | LDO_RDY | Efuse LDO ready signal<br>0: LDO is not ready<br>1: LDO is ready<br>**Note:** The signal is valid for whether the LDO bypass mode is enabled.<br>This bit is automatically set by hardware before programming and automatically cleared by hardware after programming. |
| 3 | PVIF | Program voltage setting error flag<br>0: Program voltage is in correct range<br>1: Program voltage is not in correct range |
| 2 | RDIF | Read completed flag<br>0: Read not completed<br>1: Read completed |
| 1 | PGIF | Program completed flag<br>0: Program not completed<br>1: Program completed |
| 0 | IAERRIF | Illegal access error flag<br>0: No illegal access error occurred<br>1: Illegal access error has occurred |

### 4.4.4. Status flag clear register (EFUSE_STATC)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | PVIC | RDIC | PGIC | IAERRIC |
| | | | | | | | | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | PVIC | Clear bit for program voltage setting error interrupt flag<br>0: No effect<br>1: Clear error flag |
| 2 | RDIC | Clear bit for read completed interrupt flag<br>0: No effect<br>1: Clear error flag |
| 1 | PGIC | Clear bit for program completed interrupt flag<br>0: No effect<br>1: Clear error flag |
| 0 | IAERRIC | Clear bit for illegal access error interrupt flag<br>0: No effect<br>1: Clear error flag |

### 4.4.5. User control register (EFUSE_USER_CTL)

Address offset: 0x14

Reset value: 0xXXXX XXXX. Load efuse macro values after reset.

This register can be read out without restriction. The high 16 bits of this register can be written by user only when SCRLK bit is 0. The low 16 bits of this register can be written by user only when UCLK bit is 0. But the modified value of all bits in this register will not be stored in efuse macro, unless a efuse program operation are executed successfully.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_AREA_END[7:0] | | | | | | | | SCR_AREA_START[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SCR | SPC_H | SPC_L | JTAGNSW | NDBG | | Reserved | | | UDLK | AESEN | UCLK | SCRLK | DPLK |
| | | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | SCR_AREA_END[7:0] | Secure user area end address bits<br>The factory value of these bits are 0.<br>These bits contain the last 32K-byte block of the secure user area. |

| | | |
|---|---|---|
| | | The secure user area can be defined by efuse with a granularity of 32 Kbytes.<br>End absolute address = ( SCR_AREA_END[7:0] + 1) * 32768 - 1 + 0x0800_0000.<br>If SCR_AREA_END[7:0] = SCR_AREA_START[7:0] = 0 , the secure user area is undefined.<br>If SCR_AREA_END[7:0] = SCR_AREA_START[7:0] > 0 , whole main flash block is secure user.<br>If SCR_AREA_END[7:0] < SCR_AREA_START[7:0], secure user area is invalid.<br>Refer to ***Table 3-6. Secure user area configuration*** for more details. |
| 23:16 | SCR_AREA_START[7:0] | Secure user area start address bits<br>The factory value of these bits are 0.<br>These bits contain the first 32K-byte block of the secure user area.<br>The secure user area can be defined by efuse with a granularity of 32 Kbytes.<br>Start absolute address = SCR_AREA_START[7:0] * 32768 + 0x0800_0000.<br>If SCR_AREA_END[7:0] = SCR_AREA_START[7:0] = 0 , the secure user area is undefined.<br>If SCR_AREA_END[7:0] = SCR_AREA_START[7:0] > 0, whole main flash block is secure user.<br>If SCR_AREA_END[7:0] < SCR_AREA_START[7:0], secure user area is invalid.<br>Refer to ***Table 3-6. Secure user area configuration*** for more details. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SCR | Secure mode enable<br>The factory value of this bit is 0.<br>0: Disable secure mode.<br>1: Enable secure mode.<br>**Note:** As long as this bit or SCR bit in FMC_OBSTAT0_EFT register is 1, the secure mode is enable. |
| 12 | SPC_H | Configure security protection to level high<br>The factory value of this bit is 0.<br>**Note:** If SPC_H and SPC_L in efuse macro are both 1, SPC is level high.<br>Refer to ***Table 3-4. SPC protection level configuration*** for more details. |
| 11 | SPC_L | Configure security protection to level low<br>The factory value of this bit is 0.<br>**Note:** If SPC_L in efuse macro is 1, SPC level low to no protection demotion is forbidden. If SPC_H and SPC_L in efuse macro are both set, SPC is level high.<br>Refer to ***Table 3-4. SPC protection level configuration*** for more details. |
| 10 | JTAGNSW | SW or JTAG debugger select<br>The factory value of this bit is 0.<br>0: SW<br>1: JTAG<br>**Note:** When NDBG[1:0] is selected as no debug function, JTAGNSW bit is invalid and debug function is closed. |

| | | |
|---|---|---|
| 9:8 | NDBG[1:0] | Debugging permission setting<br>The factory value of these bits are 0.<br>00: Normal JTAG ( only valid when JTAGNSW bit is 1, otherwise SW debugger is selected )<br>01: Secure JTAG ( only valid when JTAGNSW bit is 1, otherwise SW debugger is selected )<br>10~11: No debug ( debug function is closed regardless of the JTAGNSW bit ) |
| 7:5 | Reserved | Must be kept at reset value. |
| 4 | UDLK | EFUSE_USER_DATAx register lock bit<br>The factory value of this bit is 0.<br>0: Unlock EFUSE_USER_DATAx register, the register can be modified.<br>1: Lock EFUSE_USER_DATAx register, the register can not be modified. |
| 3 | AESEN | Lock EFUSE_AES_KEYx register and enable AES decrypt function<br>The factory value of this bit is 0.<br>0: Disable AES decrypt and AES key can be written<br>1: Enable AES decrypt and AES key can’t be written |
| 2 | UCLK | Low 16 bits of EFUSE_USER_CTL register lock bit<br>The factory value of this bit is 0.<br>0: Unlock the low 16 bits of EFUSE_USER_CTL register, the bits can be modified<br>1: Lock the low 16 bits of EFUSE_USER_CTL register, the bits can not be modified<br>If the UCLK bit is 1, other lock bits in EFUSE_USER_CTL register cannot be modified, so user should be careful when set UCLK bit.<br>**Note:** When UCLK bit is “1”, if user want to modify the high 16 bits of the User control parameter in efuse macro, the start address must be set to 10'd16 (EFSIZE[4:0] = 1 or 2) or 10'd24 (EFSIZE[4:0] = 1). Otherwise, IAERRIF flag will be set. |
| 1 | SCRLK | Secure userarea address lock bit<br>The factory value of this bit is 0.<br>0: Unlock the high 16 bits in EFUSE_USER_CTL register, the bits can be modified.<br>1: Lock the high 16 bits in EFUSE_USER_CTL register, the bits can not be modified.<br>**Note:** When SCRLK bit is “1”, if user want to modify the lower 16 bits of the User control parameter in efuse macro, the start address must be set to 10'd0 (EFSIZE[4:0] = 1 or 2) or 10'd8 (EFSIZE[4:0] = 1). Otherwise, IAERRIF flag will be set. |
| 0 | DPLK | EFUSE_DPx register lock bit<br>The factory value of this bit is 0.<br>0: Unlock EFUSE_DPx register, the register can be written or read.<br>1: Lock EFUSE_DPx register, the register can not be written. When this bit is 1, the JTAGNSW bit is 1, and the NDBG[1:0] bits are 2b'01 or 2b'11, the register can not be read. Otherwise this register can be read. |

### 4.4.6. MCU reserved register (EFUSE_MCU_RSV)

Address offset: 0x18

Reset value: 0xXXXX XXXX. Load efuse macro values after reset.

The register can be read out without restriction. The high 16 bits of this register can be written by user only when DCRPLK bit is 0. The low 16 bits of this register can be written by user only when MCURSVLK bit is 0. But the modified value of all bits in this register will not be stored in efuse macro, unless a efuse program operation are executed successfully.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCRP_AREA_END[7:0] | | | | | | | | DCRP_AREA_START[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MCU_RSV[5:0] | | | | | | DCRPLK | MCURSVLK | VFIMG | DISLFI | Reserved | | | | | AESNCAU |
| rw | | | | | | rw | rw | rw | rw | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DCRP_AREA_END[7:0] | DCRP area end address bits<br>The factory value of these bits are 0.<br>These bits contain the last 32K-byte block of the DCRP area.<br>The DCRP area can be defined by efuse with a granularity of 32 Kbytes.<br>End absolute address = (DCRP_AREA_END[7:0] + 1) * 32768 - 1 + 0x0800_0000.<br>If DCRP_AREA_END[7:0] = DCRP_AREA_START[7:0] = 0, DCRP area is undefined.<br>If DCRP_AREA_END[7:0] = DCRP_AREA_START[7:0] > 0, whole main flash block is DCRP area.<br>If DCRP_AREA_END[7:0] < DCRP_AREA_START[7:0], DCRP area is invalid.<br>Refer to ***Table 3-5. DCRP area configuration*** for more details. |
| 23:16 | DCRP_AREA_START[7:0] | DCRP area start address bits<br>The factory value of these bits are 0.<br>These bits contain the first 32K-byte block of the DCRP area.<br>The DCRP area can be defined by efuse with a granularity of 32 Kbytes.<br>Start absolute address = DCRP_AREA_START[7:0] * 32768 + 0x0800_0000.<br>If DCRP_AREA_END[7:0] = DCRP_AREA_START[7:0] = 0, DCRP area is undefined.<br>If DCRP_AREA_END[7:0] = DCRP_AREA_START[7:0] > 0, whole main flash block is DCRP area.<br>If DCRP_AREA_END[7:0] < DCRP_AREA_START[7:0], DCRP area is invalid.<br>Refer to ***Table 3-5. DCRP area configuration*** for more details. |
| 15:10 | MCU_RSV[5:0] | MCU reserved data |
| 9 | DCRPLK | DCRP area address lock bit<br>The factory value of this bit is 0. |

| | | |
|---|---|---|
| | | 0: Unlock the high 16 bits in EFUSE_MCU_RSV register, the bits can be modified.<br>1: Lock the high 16 bits in EFUSE_MCU_RSV register, the bits can not be modified.<br>**Note:** When DCRPLK bit is “1”, if user want to modify the lower 16 bits of the MCU reserved parameter in efuse macro, the start address must be set to 10'd32 (EFSIZE[4:0] = 1 or 2) or 10'd40 (EFSIZE[4:0] = 1). Otherwise, IAERRIF flag will be set. |
| 8 | MCURSVLK | Low 16 bits of EFUSE_MCU_RSV register lock bit<br>The factory value of this bit is 0.<br>0: Unlock the low 16 bits of EFUSE_MCU_RSV register, the bits can be modified<br>1: Lock the low 16 bits of EFUSE_MCU_RSV register, the bits can not be modified<br>If the MCURSVLK bit is 1, other lock bits in EFUSE_MCU_RSV register cannot be modified, so user should be careful when set MCURSVLK bit.<br>**Note:** When the MCURSVLK bit is “1”, If user want to modify the high 16 bits of the MCU reserved parameter in efuse macro, the start address must be set to 10'd48 (EFSIZE[4:0] = 1 or 2) or 10'd56 (EFSIZE[4:0] = 1). Otherwise, IAERRIF flag will be set. |
| 7 | VFIMG | Firmware image verification enable bit<br>The factory value of this bit is 0.<br>0: Disable firmware image verification<br>1: Enable firmware image verification |
| 6 | DISLFI | Licensed firmware install ( LFI ) disable<br>The factory value of this bit is 0.<br>0: Enable licensed firmware install<br>1: Disable licensed firmware install |
| 5:1 | Reserved | Must be kept at reset value. |
| 0 | AESNCAU | AES key for cau configuration bit (AESNCAU only for GD32H737_757_759)<br>The factory value of this bit is 0.<br>0: The AES key can be used for CAU<br>1: The AES key cannot be used for CAU |

### 4.4.7. Debug password register x (EFUSE_DPx) (x = 0,1)

Address offset: 0x1C + 0x4 * x
Reset value: 0xXXXX XXXX. Load efuse macro values after reset.

When JTAGNSW = 1, and NDBG[1:0] = 2b'01 or 2b'11, this parameter is used as the debug password for debug services. Otherwise, this parameter will be used as user data.

Used as debug password: this register can be read out only when DPLK is 0. The register can be written only when DPLK bit is 0. But the modified value of register will not be stored in efuse macro, unless a efuse program operation are executed successfully.

Used as user data: the register can be read regardless of DPLK bit value. The register can be written only when DPLK bit is 0. But the modified value of register will not be stored in efuse macro, unless a efuse program operation are executed successfully.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DP[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DP[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DPx[31:0] | Efuse debug password value.<br>The factory value of these bits are 0. |

### 4.4.8. Firmware AES key register x (EFUSE_AES_KEYx) (x = 0…3)

Address offset: 0x24 + 0x4 * x
Reset value: 0xXXXX XXXX. Load efuse macro values after reset.

This register cannot be read. This register can be written only when AESEN is 0. But the modified value of register will not be stored in efuse macro, unless a efuse program operation are executed successfully.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AESKEY[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AESKEY[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | AESKEY[31:0] | Efuse AES key value.<br>The factory value of these bits are 0.<br>User must continuously write the complete 16 bytes AES key into the EFUSE_AES_KEYx registers.<br>The register can only be accessed by a word (32-bit) write. The 4 bytes in each register are stored in order from low to high (that is, the lower byte of the AESKEY corresponds to the lower bits of the register).<br>Meanwhile, AESKEY[31:0] writes to the EFUSE_AES_KEY0 register (offset 0x24), AESKEY[63:32] writes to the EFUSE_AES_KEY1 register (offset 0x28), AESKEY[95:64] writes to the EFUSE_AES_KEY2 register (offset 0x2C) and AESKEY[127:96] writes to the EFUSE_AES_KEY3 register (offset 0x30). |

CRC calculation result is generated after writing the EFUSE_AES_KEY3 register (offset address 0x30).

### 4.4.9. User data register x (EFUSE_USER_DATAx) (x = 0…3)

Address offset: 0x34 + 0x4 * x

Reset value: 0xXXXX XXXX. Load efuse macro values after reset.

This register can be read out without restriction. This register can be modified only when UDLK is 0. This register can be written only when UDLK is 0. But the modified value of register will not be stored in efuse macro, unless a efuse program operation are executed successfully.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USERDATA[31:16] | | | | | | | | | | | | | | | |
| Rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| USERDATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | USERDATA[31:0] | Efuse USER_DATA value.<br>The factory value of these bits are 0. |

# 5. Power management unit (PMU)

## 5.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32H73x_75x series. The Power management unit (PMU) provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. These modes reduce the power consumption and allow the application to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. For GD32H73x_75x devices, there are three power domains, including $V_{DD}$ / $V_{DDA}$ domain, $V_{CORE}$ domain, and Backup domain, as is shown in ***Figure 5-1. Power supply overview***. The power of the $V_{DD}$ domain is supplied directly by $V_{DD}$. An embedded LDO and Step-down voltage stabilizer, which is low power switched-mode power supply (SMPS step-down voltage stabilizer) is used to supply the $V_{CORE}$ domain power. A power switch is implemented for the Backup domain. It can be powered from the $V_{BAT}$ voltage when the main $V_{DD}$ supply is shut down. Peripheral supply regulation USB regulator.

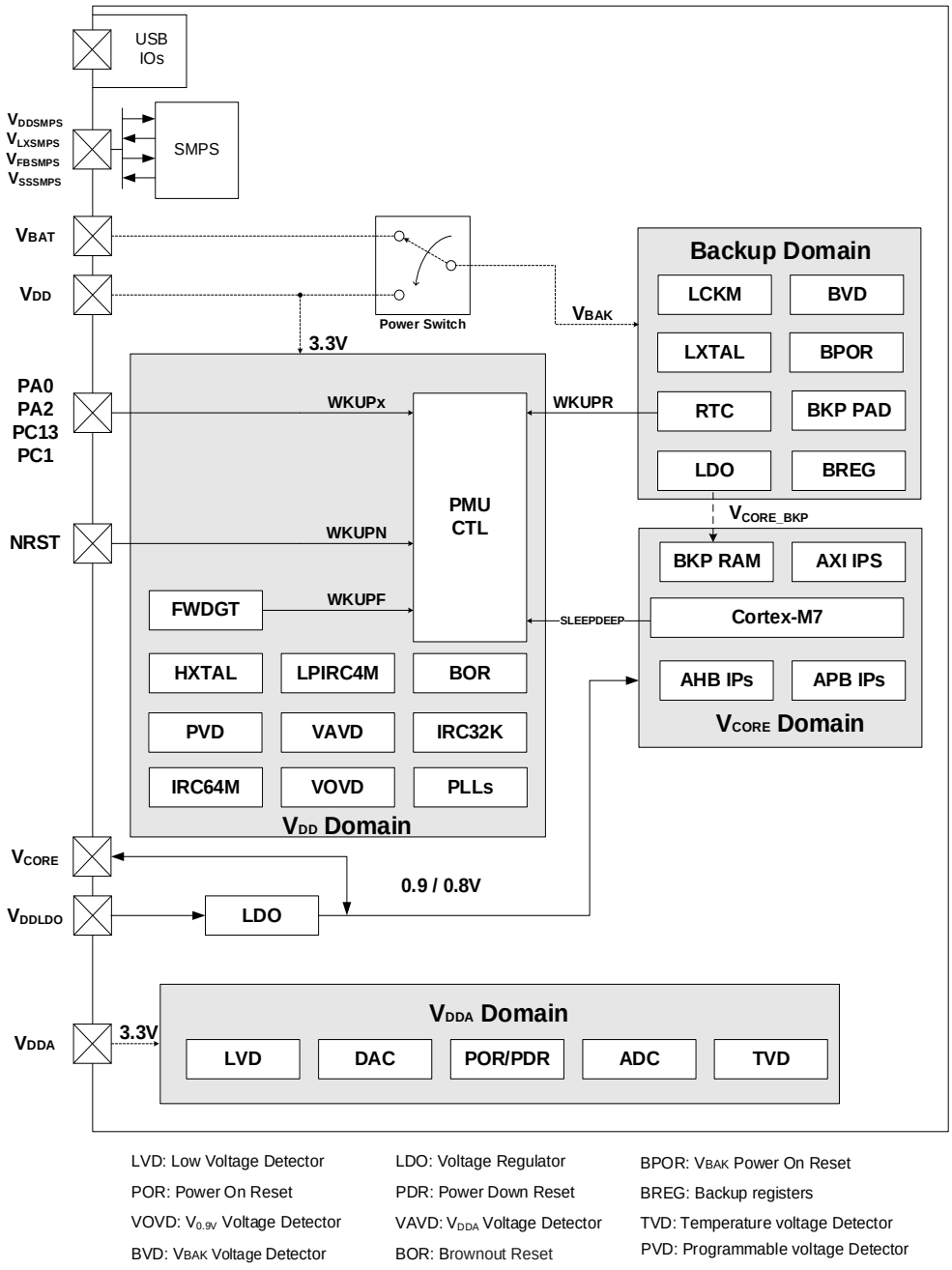
## 5.2. Characteristics

- Three power domains: $V_{BAK}$, $V_{DD}$ / $V_{DDA}$ and $V_{CORE}$ power domains.
- Three power saving modes: Sleep, Deep-sleep and Standby modes.
- Internal Voltage regulator (LDO) supplies around $V_{CORE}$ voltage source for $V_{CORE}$ domain.
- Low Voltage Detector (LVD) can issue an interrupt or event when the power is lower than a programmed threshold.
- Battery power ($V_{BAT}$) for Backup domain when $V_{DD}$ is shut down.
- LDO output voltage select for power saving.
- USB supply regulator.
- Power supply supervision: POR / PDR monitor / BOR monitor / LVD monitor / $V_{DDA}$ Voltage Detector (VAVD) monitor / $V_{BAK}$ thresholds / Temperature thresholds.
- $V_{BAT}$ battery charging / Operating modes / Voltage scaling control / Low-power modes management.
- Step-down voltage stabilizer, which is low power switched-mode power supply (SMPS step-down voltage stabilizer).
- Supports the status output for the CPU entering deep sleep or sleep mode.

## 5.3. Function overview

*Figure 5-1. Power supply overview* provides details on the internal configuration of the PMU and the relevant power domains.

**Figure 5-1. Power supply overview**




**Note**: The SMPS power supply is not supported on all devices. Please refer to the datasheet for details.

### 5.3.1. Backup domain

The Backup domain is powered by the $V_{DD}$ or the battery power source ($V_{BAT}$) selected by the internal power switch, and the $V_{BAK}$ pin which drives Backup domain, supplies power for RTC unit, LXTAL oscillator, LCKM (LXTAL clock monitor), BVD, BPOR and BREG, and three BKP PADs, including PC13 to PC15. In order to ensure the content of the Backup domain registers and the RTC supply, when $V_{DD}$ supply is shut down, $V_{BAT}$ pin can be connected to an optional standby voltage supplied by a battery or by another source. The power switch is controlled by the power down reset circuit in the $V_{DD}$ / $V_{DDA}$ domain. If no external battery is used in the application, it is recommended to connect $V_{BAT}$ pin externally to $V_{DD}$ pin with a 100nF external ceramic decoupling capacitor.

The Backup domain reset sources includes the Backup domain power-on-reset (BPOR) and the Backup domain software reset. The BPOR signal forces the device to stay in the reset mode until $V_{BAK}$ is completely powered up. Also the application software can trigger the Backup domain software reset by setting the BKPRST bit in the RCU_BDCTL register to reset the Backup domain.

The clock source of the Real Time Clock (RTC) circuit can be derived from the Internal 32KHz RC oscillator (IRC32K) or the Low Speed Crystal oscillator (LXTAL), or HXTAL divided by RTCDIV[5:0] (in RCU_CFG0 register). When $V_{DD}$ is shut down, only LXTAL is valid for RTC. Before entering the power saving mode by executing the WFI / WFE instruction, the Cortex®-M7 can setup the RTC register with an expected alarm time and enable the alarm function and according EXTI lines to achieve the RTC alarm event. After entering the power saving mode for a certain amount of time, the RTC alarm will wake up the device when the time match event occurs. The details of the RTC configuration and operation will be described in the ***Real time clock (RTC)***.

When the Backup domain is supplied by $V_{DD}$ ($V_{BAK}$ pin is connected to $V_{DD}$), the following functions are available:

- PC13 can be used as GPIO or RTC function pin described in ***Real time clock (RTC)***.
- PC14 and PC15 can be used as either GPIO or LXTAL Crystal oscillator pins.

When the Backup domain is supplied by $V_{BAT}$ ($V_{BAK}$ pin is connected to $V_{BAT}$), the following functions are available:

- PC13 can be used as RTC function pin described in the ***Real time clock (RTC)*** chapter.
- PC14 and PC15 can be used as LXTAL Crystal oscillator pins only.

**Note:** Since PC13, PC14, PC15 are supplied through the power switch, which can only be obtained by a small current, the speed of GPIOs PC13 to PC15 should not exceed 2MHz when they are in output mode (maximum load: 30pF).

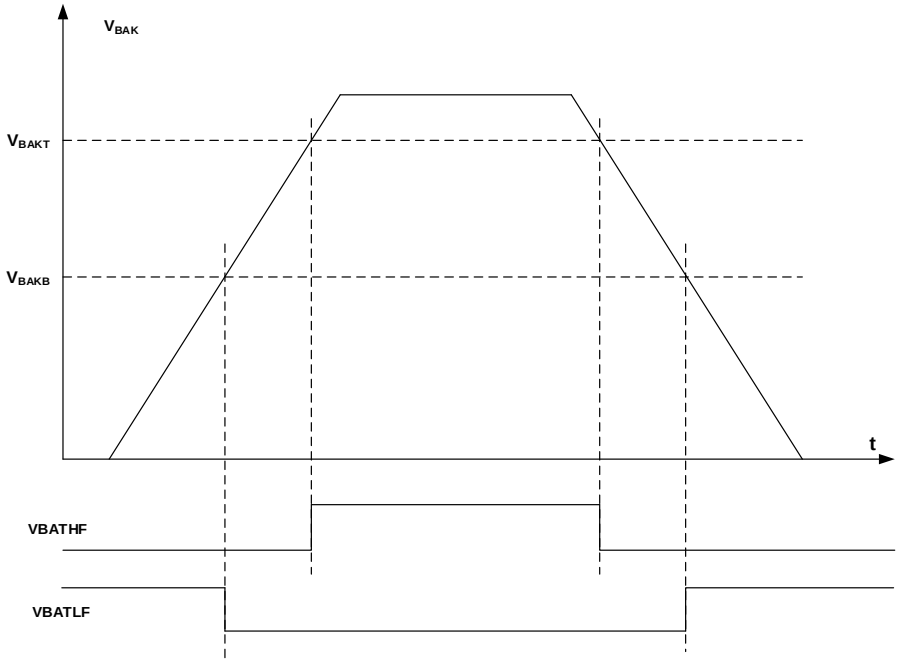
The external $V_{BAT}$ battery can be charged by the $V_{DD}$ through an internal resistor. The charging resistor can be selected by configuring the VCRSEL bit in PMU_CTL2 register. A 5kOhms resistor or a 1.5kOhms resistor can be selected for external $V_{BAT}$ battery charing. The external $V_{BAT}$ battery charing is enabled by setting the VCEN bit in PMU_CTL2 register. When in BKP only mode, the $V_{BAT}$ battery charing is disabled by hardware.

**Note:** In BKP only mode, $V_{DD}$ is power-off, and the backup domain is power by $V_{BAT}$ pin.

### Backup domain voltage thresholds

There is an internal power switch, which can select the voltage source of Backup domain $V_{BAT}$ or $V_{DD}$ The supply voltage for Backup domain ($V_{BAK}$) can be monitored with a top voltage and a bottom voltage ($V_{BAKT}$ and $V_{BAKB}$), when VBTMEN bit is set, if $V_{BAK}$ is over $V_{BAKT}$ or lower than $V_{BAKB}$, the flag bit $V_{BATHF}$ / $V_{BATLF}$ will set. And this is only available when BKPVSEN bit is set. Backup domain voltage thresholds, as is shown in ***Figure 5-2. Waveform of the Backup domain voltage thresholds***.

**Figure 5-2. Waveform of the Backup domain voltage thresholds**




## 5.3.2. $V_{DD}$ / $V_{DDA}$ power domain

$V_{DD}$ / $V_{DDA}$ domain includes two parts: $V_{DD}$ domain and $V_{DDA}$ domain. $V_{DD}$ domain includes HXTAL (high speed crystal oscillator), FWDGT (free watchdog timer), IRC64M (internal 64MHz RC oscillator at 64MHz frequency), LPIRC4M (internal 4MHz RC oscillator), BOR (Brownout Reset), PLLs (phase locking loop), VOVD ($V_{CORE}$ voltage detector), VAVD ($V_{DDA}$ voltage detector), PVD (Programmable voltage detector), IRC32K (internal 32KHz RC oscillator), all pads except PC13 / PC14 / PC15, etc. $V_{DDA}$ domain includes ADC / DAC (AD / DA Converter), POR / PDR (power on / down reset), LVD (low voltage detector), TVD (temperature voltage detector), etc.

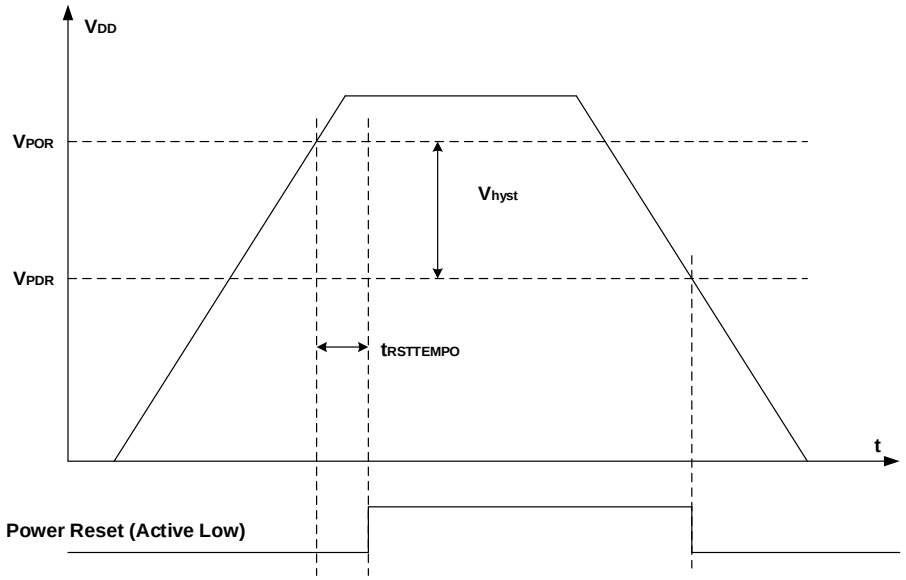
### $V_{DD}$ domain

The LDO, which is implemented to supply power for the $V_{CORE}$ domain, is always enabled after reset. It can be configured to operate in different status, including in the Sleep mode ($V_{CORE}$ full power on, low power), in the Deep-sleep mode (on or low power), and in the

Standby mode (power off).

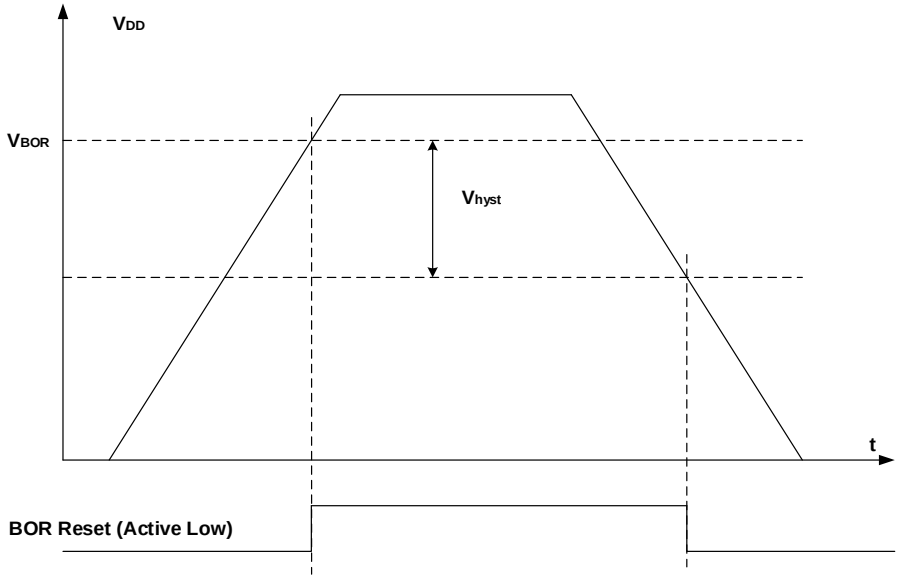
The POR / PDR circuit is implemented to detect $V_{DD}$ and generate the power reset signal which resets the whole chip except the Backup domain when the supply voltage is lower than the specified threshold. ***Figure 5-3. Waveform of the POR / PDR*** shows the relationship between the supply voltage and the power reset signal. $V_{POR}$ indicates the threshold of power on reset, while $V_{PDR}$ means the threshold of power down reset. The hysteresis voltage (Vhyst) is refer to the datasheet.

**Figure 5-3. Waveform of the POR / PDR**




The BOR circuit is used to detect $V_{DD}$ and generate the power reset signal which resets the whole chip except the Backup domain when the BOR_TH bits in option bytes is not 0b00 and the supply voltage is lower than the specified threshold which defined in the BOR_TH bits in option bytes. Notice that the POR / PDR circuit is always implemented regardless of BOR_TH bits in option bytes is 0b00 or not. ***Figure 5-4. Waveform of the BOR*** shows the relationship between the supply voltage and the BOR reset signal. $V_{BOR}$, which defined in the BOR_TH bits in option bytes, indicates the threshold of BOR on reset. The hysteresis voltage ($V_{hyst}$) is refer to the datasheet.

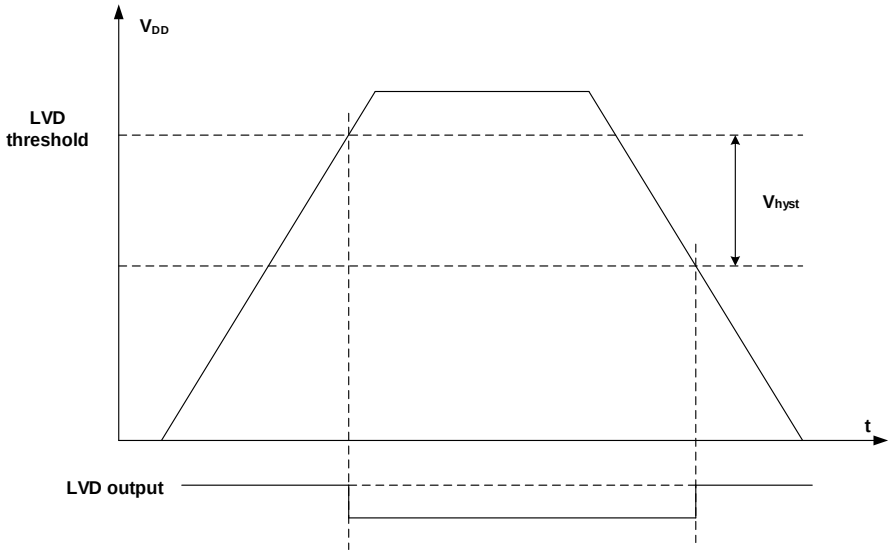
**Figure 5-4. Waveform of the BOR**




## $V_{DDA}$ domain

The LVD is used to detect whether the $V_{DD}$ supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU_CTL0). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in PMU_CS, indicates if $V_{DD}$ is higher or lower than the LVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers.***Figure 5-5. Waveform of the LVD threshold*** shows the relationship between the LVD threshold and the LVD output (LVD interrupt signal depends on EXTI line 16 rising or falling edge configuration). The following figure shows the relationship between the supply voltage and the LVD signal. The hysteresis voltage ($V_{hyst}$) is refer to the datasheet.

**Figure 5-5. Waveform of the LVD threshold**




Generally, digital circuits are powered by $V_{DD}$, while most of analog circuits are powered by

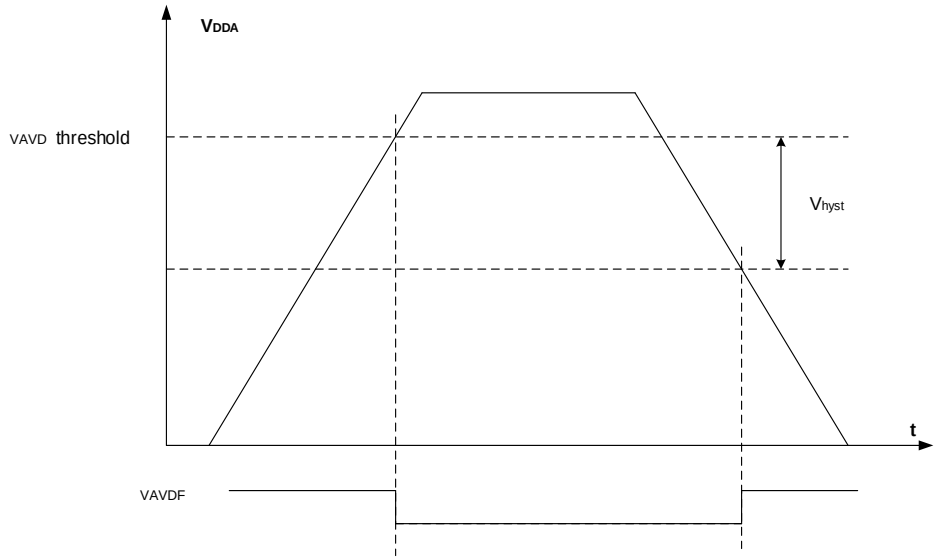
$V_{DDA}$. To improve the ADC and DAC conversion accuracy, the independent power supply $V_{DDA}$ is implemented to achieve better performance of analog circuits. $V_{DDA}$ can be externally connected to $V_{DD}$ through the external filtering circuit that avoids noise on $V_{DDA}$, and $V_{SSA}$ should be connected to $V_{SS}$ through the specific circuit independently. Otherwise, when the $V_{DD}$ and $V_{DDA}$ are provided by different power supplies, the difference between $V_{DD}$ and $V_{DDA}$ during power-up and running time should not exceed 0.3V.

To ensure a high accuracy on ADC and DAC, the ADC / DAC independent external reference voltage should be connected to VREFP / VREFN pins. According to the different packages, VREFP pin can be connected to $V_{DDA}$ pin, or external reference voltage which refers to ***Table 20-2. ADC input pins definition*** and ***Table 21-1. DAC I/O description***, VREFN pin must be connected to $V_{SSA}$ pin. The VREFP pin is only available on no less than 100-pin packages, or else the VREFP pin is not available and internally connected to $V_{DDA}$. The VREFN pin is only available on no less than 100-pin packages, or else the VREFN pin is not available and internally connected to $V_{SSA}$.

### $V_{DDA}$ domain voltage thresholds

The $V_{DDA}$ analog voltage detector (VAVD) is used to detect whether the $V_{DDA}$ supply voltage is lower than a programmed threshold selected by the VAVDVC[1:0] bits in the power control register(PMU_CTL0). The VAVD is enabled by setting the VAVDEN bit, and VAVDF bit, which in PMU_CS, indicates if $V_{DDA}$ is higher or lower than the specified VAVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers. ***Figure 5-6. Waveform of the VAVD threshold*** shows the relationship between the VAVD threshold and the VAVDF. The hysteresis voltage (Vhyst) is refer to the datasheet.

**Figure 5-6. Waveform of the VAVD threshold**



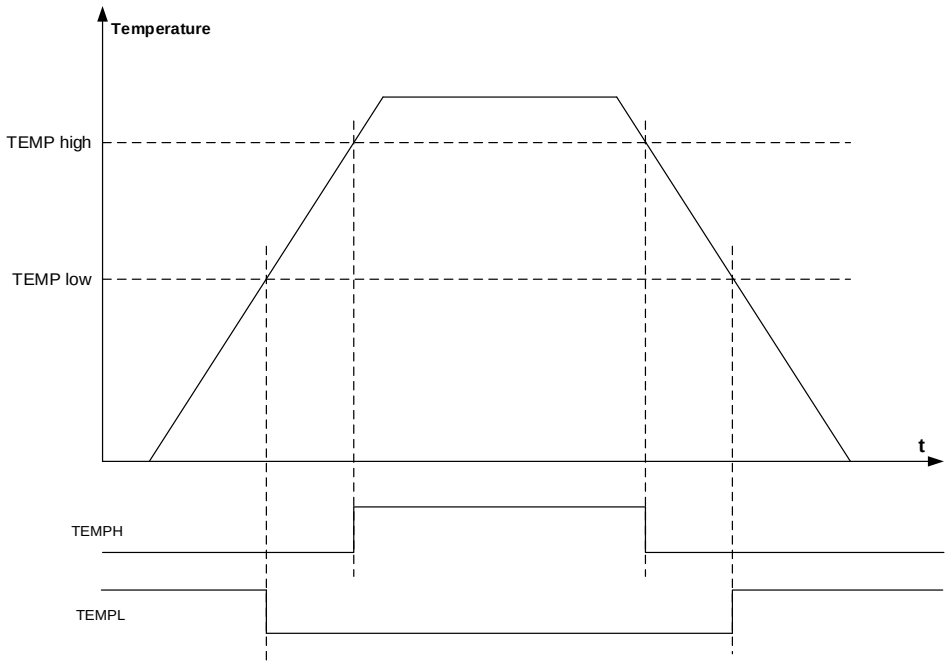

### Temperature voltage thresholds

As will as Backup domain voltage thresholds, The junction temperature can be monitored by

comparing it with two threshold levels, TEMP high and TEMP low. TEMPH and TEMPL flags, in the PMU_CTL1, indicate whether the device temperature is higher or lower than the threshold. The temperature voltage thresholds can be enabled / disabled via VBTMEN bit in PMU_CTL1. When enabled, the temperature thresholds increase power consumption. As an example the levels could be used to trigger a routine to perform temperature control tasks. The temperature thresholds are available only when the backup regulator is enabled (VBTMEN it set in the PMU_CTL1 register).

TEMPH and TEMPL wakeup interrupts are available on the RTC tamper signals .

**Figure 5-7. Temperature thresholds**




### 5.3.3. $V_{CORE}$ power domain

The main functions that include Cortex®-M7 logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the $V_{DD}$ / $V_{DDA}$ domain, etc, are located in this power domain. Once the $V_{CORE}$ is powered up, the POR will generate a reset sequence on the $V_{CORE}$ power domain. If need to enter the expected power saving mode, the associated control bits must be configured. Then, once a WFI (Wait for Interrupt) or WFE (Wait for Event) instruction is executed, the device will enter an expected power saving mode.

#### $V_{CORE}$ power supply

With the SMPS step-down stabilizer and LDO, it is possible to set the power source of the $V_{CORE}$ power domain. Different configurations can provide four effective $V_{CORE}$ power supply modes.

- No configuration power supplies mode

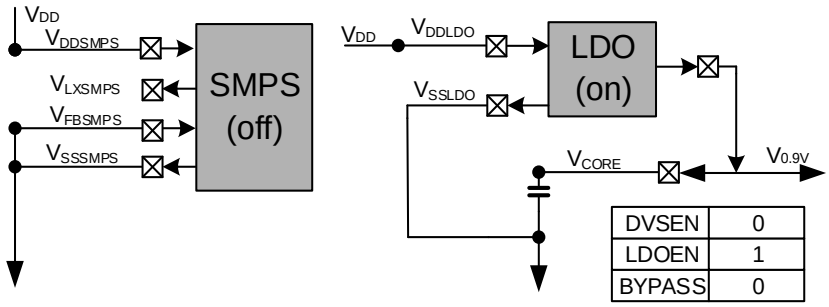
After reset, the DVSEN bit is 0b1. At this time, the SMPS step-down stabilizer is on, working

in normal mode, working voltage is 1.0V, and SMPS step-down stabilizer can power supplies for LDO; LDOEN bit is 0b1, LDO is on, and power supplies for $V_{CORE}$ power domain, the power supplies voltage is controlled by LDOVS[2:0] bits field; BYPASS bit is 0b0, $V_{CORE}$ power domain is not powered by $V_{CORE}$ (external direct power supplies).

- LDO power supplies mode

The configuration method to enter this mode is that the DVSEN bit is 0b0. At this time, the SMPS step-down stabilizer is in the off state; the LDOEN bit is 0b1, and the LDO is on. It also supplies power for the $V_{CORE}$ power domain. The power supplies voltage is controlled by the LDOVS[2:0] bits. The working mode of the LDO is consistent with the low power consumption mode of the system; the BYPASS bit is 0b0, and the $V_{CORE}$ power domain is not powered through $V_{CORE}$ (external direct power supplies ). ***Figure 5-8. LDO supplies for VCORE power domain*** shows this power supplies mode.

**Figure 5-8. LDO supplies for $V_{CORE}$ power domain**

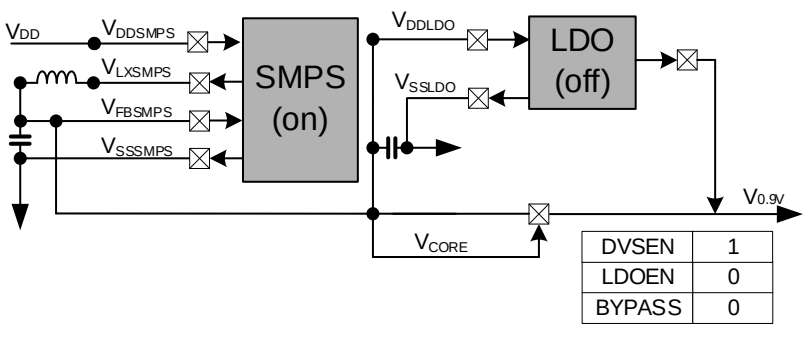



LDO supply

- SMPS power supplies mode

The configuration method to enter this mode is that the DVSEN bit is 0b1. At this time, the SMPS step-down stabilizer is in the on state, It also supplies power for the $V_{CORE}$ power domain. The power supplies voltage is controlled by the LDOVS[2:0] bits, the working mode of the SMPS is consistent with the low power consumption mode of the system; the LDOEN bit is 0b0, and the LDO is off; the BYPASS bit is 0b0, and the $V_{CORE}$ power domain is not powered through $V_{CORE}$ (external direct power supplies ). ***Figure 5-9. SMPS supplies for VCORE power domain*** shows this power supplies mode.

**Figure 5-9. SMPS supplies for $V_{CORE}$ power domain**




SMPS supply

- bypass mode

The configuration method to enter this mode is that the DVSEN bit is 0b0. At this time, the SMPS step-down stabilizer is in the off; the LDOEN bit is 0b0, and the LDO is off; the BYPASS bit is 0b1, and the $V_{CORE}$ power domain is powered through $V_{CORE}$ (external direct power supplies). ***Figure 5-10. Bypass*** shows this power supplies mode.

**Figure 5-10. Bypass**

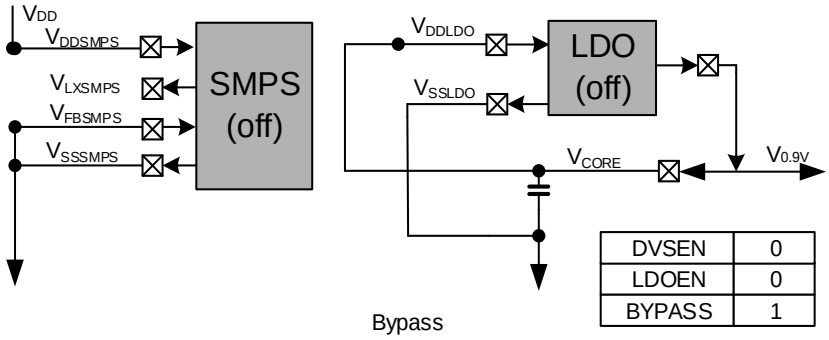



**Table 5-1. Supply mode**

| Mode | Supply configuration | DVSEN | LDOEN | LDOVS | BYPASS |
|---|---|---|---|---|---|
| 0 | No configuration power supplies mode | 1 | 1 | 0b010 | 0 |
| 1 | LDO power supplies mode | 0 | 1 | 0b000-101 | 0 |
| 2 | SMPS power supplies mode | 1 | 0 | 0b000-0b101 | 0 |
| 6 | bypass mode | 0 | 0 | x | 1 |

**Note:** In addition to the above valid combinations, other DVSEN, LDOEN, BYPASS bits or bit field configuration combinations are invalid. The power supplies state of the $V_{CORE}$ power domain will remain the state after reset (no configure the power supplies mode).

**Note:** The maximum working frequency is related to the supply voltage. Please refer to the datasheet for details.

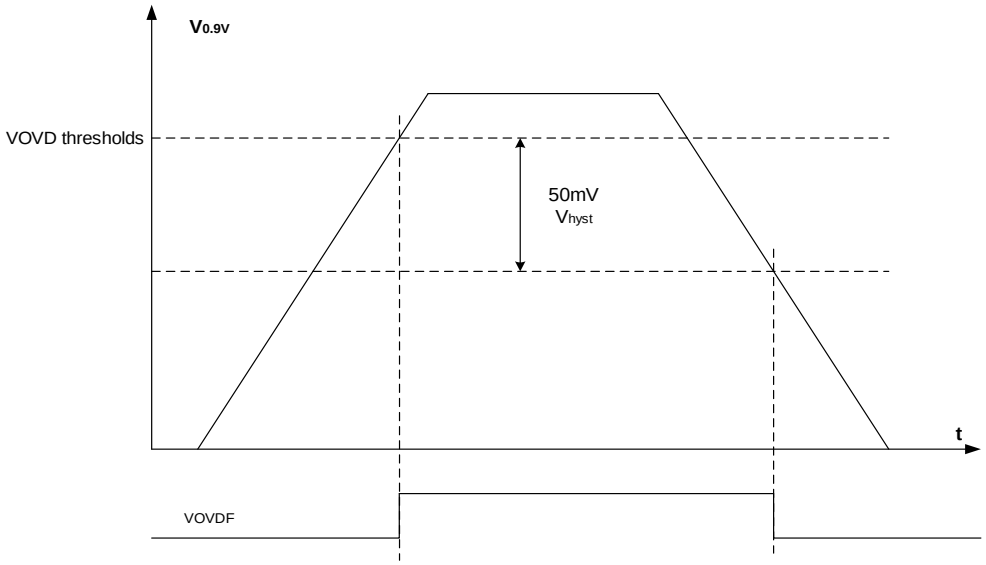
**Note**: The SMPS power supply is not supported on all devices. Please refer to the datasheet for details.

**Note**: The use of bypass mode. Please refer to the **AN225 GD32H7xx Power Bypass Mode User Guide** for details.

### $V_{CORE}$ power thresholds detector

There is an internal $V_{CORE}$ power thresholds detector, when VOVDEN is 0b1, it means $V_{CORE}$ power thresholds detector is enabled. Once $V_{CORE}$ power domain is over voltage, VOVDF will be set.

**Figure 5-11. Waveform of the VOVD**




## 5.3.4. USB Power supply

The GD32H73x_75x series USB integrates a voltage regulator internally, which users can choose to enable. Connect the VDD50USB pin to a 5V power supply to provide power for the USB module, as shown in ***Figure 5-12. Connection diagram of USB voltage regulator during power supply***; Alternatively, bypass the voltage regulator and connect the VDD33USB pin to a 3.3V power supply to provide power to the USB module, as shown in ***Figure 5-13. Connection diagram for USB voltage regulator bypass***.

**Figure 5-12. Connection diagram of USB voltage regulator during power supply**

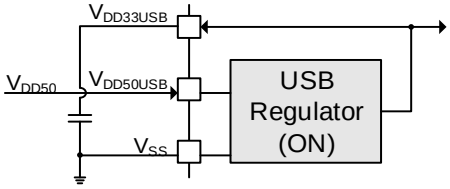
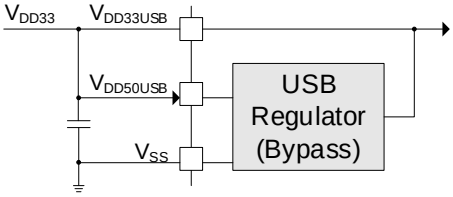



**Figure 5-13. Connection diagram for USB voltage regulator bypass**




**Note:** When VDD ≥ 3V, the USB power can use the USB regulator power supply mode or bypass mode; when VDD < 3V, the USB power can only use the USB regulator power supply mode.

## 5.3.5. Power saving modes

After a system reset or a power reset, the GD32H73x_75x MCU operates at full function and

all power domains are active. Users can achieve lower power consumption through slowing down the system clocks (HCLK, PCLK1, and PCLK2) or gating the clocks of the unused peripherals or configuring the LDO output voltage by LDOVS[2:0] bits in PMU_CTL3 register. The LDOVS[2:0] bits should be configured only when the PLL is off. Besides, three power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode and Standby mode.

After system reset / power reset or wakeup from standby mode. The MCU enter normal run mode, no effect on all clocks, all power on. And the LDO work in $V_{CORE}$ mode.

### Sleep mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex®-M7. In Sleep mode, only clock of Cortex®-M7 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex®-M7 System Control Register, and execute a WFI or a SEV and two WFE instruction. If the Sleep mode is entered by executing a WFI instruction, any interrupt can wake up the system. If it is entered by executing SEV and WFE instruction, any wakeup event can wake up the system (If SEVONPEND is 1, any interrupt can wake up the system, refer to Cortex-M7 Technical Reference Manual). The mode offers the lowest wakeup time as no time is wasted in interrupt entry or exit.

According to the SLEEPONEXIT bit in the Cortex®-M7 System Control Register, there are two options to select the Sleep mode entry mechanism.

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as APB system reset, WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits from the lowest priority ISR.

#### Deep-sleep mode

The Deep-sleep mode is based on the SLEEPDEEP mode of the Cortex®-M7. In Deep-sleep mode, all clocks in the $V_{CORE}$ domain are off, and all of LPIRC4M, IRC64M, HXTAL and PLLs are disabled. Before entering the Deep-sleep mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M7 System Control Register, set LPMOD bit to 0b1 in the PMU_CTL0 register. Then, the device enters the Deep-sleep mode after a WFI or WFE instruction is executed. If the Deep-sleep mode is entered by executing a WFI instruction, any interrupt from EXTI lines can wake up the system. If it is entered by executing a WFE instruction, any wakeup event from EXTI lines can wake up the system (If SEVONPEND is 0b1, any interrupt from EXTI lines can wake up the system, refer to Cortex®-M7 Technical Reference Manual).

**Note:** If power-on or exit from standby, it need to wait a moment before entering Deep-sleep mode.

### Standby mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex®-M7, too. In Standby mode, the whole $V_{CORE}$ domain is power off, the LDO is shut down, and all of LPIRC4M,

IRC64M, HXTAL and PLLs are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M7 System Control Register, and set the LPMOD bit to 0b in the PMU_CTL0 register, and clear WUF bit in the PMU_CS register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed, and the STBF status flag in the PMU_CS register indicates that the MCU has been in Standby mode. There are five wakeup sources for the Standby mode, including the external reset from NRST pin, the RTC (alarm and tamp), the FWDGT reset, LCKMD and the rising edge on WKUP pins. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers in $V_{CORE}$ power domain are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the Cortex®-M7 will execute instruction code from the 0x00000000 address.

**Table 5-2. Power saving mode summary**

| Mode | Description | LDO | Entry | Wakeup | Wakeup status | Wakeup Latency |
|---|---|---|---|---|---|---|
| Sleep | Only CPU clock is off | LDO on | SLEEPDEEP = 0, WFI or SEV and WFE from run | Any interrupt for WFI<br>Any event (or interrupt when SEVONPEND is 1) for SEV and WFE | normal run mode | - |
| Deep-sleep | 1. All clocks in the $V_{CORE}$ domain are off<br>2. Disable LPIRC4M, IRC64M, HXTAL and PLLs | LDO on | SLEEPDEEP = 1, LPMOD = 0, WFI or WFE | Any interrupt from EXTI lines for WFI<br>Any event(or interrupt when SEVONPEND is 1) from EXTI for WFE | normal run mode | LPIRC4M / IRC64M (confirmed by DSPWUSSEL) wakeup time, + Flash wakeup time |
| Standby | 1. The $V_{CORE}$ domain is power off<br>2. LPIRC4M, IRC64M, HXTAL and PLLs | LDO off | SLEEPDEEP = 1, LPMOD = 1, WFI or WFE | 1. NRST pin<br>2. WKUP pins<br>3. FWDGT reset<br>4. RTC (alarm and tamp)<br>5. LCKMD | normal run mode | IRC64M wakeup time, + LDO wakeup time+Flash wakeup time |
| BKP only mode | All VDD / $V_{CORE}$ domain power off | LDO off | VDD off | VDD on | normal run mode | VDD power on sequence |

**Note:** In Standby mode, all I/Os are in high-impedance state except NRST pin, PC13 pin when configured for RTC function, PC14 and PC15 pins when used as LXTAL crystal oscillator pins, and WKUP pins if enabled.

## 5.4. Register definition

PMU base address: 0x5800 5800

### 5.4.1. Control register 0 (PMU_CTL0)

Address offset: 0x00

Reset value: 0x0000 8000 (reset by wakeup from Standby mode)

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | VOVDEN | VAVDVC[1:0] | | VAVDEN |
| | | | | | | | | | | | | rw | rw | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SLDOVS[1:0] | | Reserved | | | | | BKPWEN | LVDT[2:0] | | | LVDEN | STBRST | WURST | STBMOD | Reserved |
| rs | | | | | | | rw | rw | | | rw | rc_w1 | rc_w1 | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value. |
| 19 | VOVDEN | Peripheral voltage on $V_{CORE}$ detector enable bit<br>This bit is set and cleared by software.<br>0: Peripheral voltage on $V_{CORE}$ detector disabled.<br>1: Peripheral voltage on $V_{CORE}$ detector enabled |
| 18:17 | VAVDVC[1:0] | $V_{DDA}$ analog voltage detector voltage level configure bits<br>These bits are set and cleared by software.<br>00: Configure $V_{DDA}$ analog voltage detector voltage level to 1.7V<br>01: Configure $V_{DDA}$ analog voltage detector voltage level to 2.1V<br>10: Configure $V_{DDA}$ analog voltage detector voltage level to 2.5V<br>11: Configure $V_{DDA}$ analog voltage detector voltage level to 2.8V |
| 16 | VAVDEN | $V_{DDA}$ analog voltage detector voltage enable bit<br>This bit is set and cleared by software.<br>0: $V_{DDA}$ analog voltage detector voltage disabled.<br>1: $V_{DDA}$ analog voltage detector voltage enabled |
| 15:14 | SLDOVS[1:0] | Deep-sleep mode voltage scaling selection<br>These bits control the $V_{CORE}$ voltage level in system Deep-sleep mode, to obtain the best trade-off between power consumption and performance.<br>00: SLDOVS Scale 0.6V<br>01: SLDOVS Scale 0.7V<br>10: SLDOVS Scale 0.8V (default)<br>11: SLDOVS Scale 0.9V |
| 13:9 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 8 | BKPWEN | Backup domain write enable bit<br>This bit is set and cleared by software.<br>0: Disable write access to the registers in backup domain<br>1: Enable write access to the registers in backup domain<br>After reset, any write access to the registers in Backup domain is ignored. This bit has to be set to enable write access to these registers. |
| 7:5 | LVDT[2:0] | Low voltage detector threshold<br>000: 2.1V<br>001: 2.3V<br>010: 2.4V<br>011: 2.6V<br>100: 2.7V<br>101: 2.9V<br>110: 3.0V<br>111: Input analog voltage on PB7 (compared with 0.8V).<br>**Note:** For measured values, please refer to the datasheet. |
| 4 | LVDEN | Low voltage detector enable bit<br>This bit is set and cleared by software.<br>0: Disable low voltage detector<br>1: Enable low voltage detector<br>**Note:** When LVD_LOCK bit is set to 1 in the SYSCFG_LKCTL register, LVDEN and LVDT[2:0] are read only. |
| 3 | STBRST | Standby flag reset bit<br>This bit is set by software.<br>0: No effect<br>1: Reset the standby flag<br>This bit is always read as 0. |
| 2 | WURST | Wakeup flag reset bit<br>This bit is set by software.<br>0: No effect<br>1: Reset the wakeup flag<br>This bit is always read as 0. |
| 1 | STBMOD | Standby Mode<br>0: Enter the Deep-sleep mode when the Cortex®-M7 enters SLEEPDEEP mode<br>1: Enter the Standby mode when the Cortex®-M7 enters SLEEPDEEP mode |
| 0 | Reserved | Must be kept at reset value. |

### 5.4.2. Control and status register (PMU_CS)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby mode)

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | VOVDF | Reserved | | | VAVDF |
| | | | | | | | | | | | r | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | WUPEN5 | Reserved | WUPEN3 | Reserved | WUPEN1 | WUPEN0 | Reserved | | | | | LVDF | STBF | WUF |
| | | rw | | rw | | rw | rw | | | | | | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | VOVDF | Peripheral voltage on $V_{CORE}$ detector flag bit<br>This bit is set and cleared by hardware. It is valid only if VOVDEN is enabled.<br>0: $V_{CORE}$ is lower than VOVD threshold(1.15V).<br>1: $V_{CORE}$ is equal or higher than VOVD threshold(1.15V) |
| 19:17 | Reserved | Must be kept at reset value. |
| 16 | VAVDF | $V_{DDA}$ analog voltage detector voltage output on $V_{DDA}$ flag bit<br>This bit is set and cleared by hardware. It is valid only if VAVDEN is enabled.<br>0: $V_{DDA}$ is equal or higher than the VAVD threshold configured by VAVDVC bits.<br>1: $V_{DDA}$ is lower than the VAVD threshold configured by VAVDVC bits |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | WUPEN5 | WKUP Pin5 (PC1) enable<br>0: Disable WKUP pin5 function<br>1: Enable WKUP pin5 function<br>If WUPEN5 is set before entering the power saving mode, a rising edge on the WKUP pin5 wakes up the system from the power saving mode. As the WKUP pin5 is active high, the WKUP pin5 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 12 | Reserved | Must be kept at reset value. |
| 11 | WUPEN3 | WKUP pin3 (PC13) enable<br>0: Disable WKUP pin3 function<br>1: Enable WKUP pin3 function<br>If WUPEN3 is set before entering the power saving mode, a rising edge on the WKUP pin3 wakes up the system from the power saving mode. As the WKUP pin3 is active high, the WKUP pin3 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 10 | Reserved | Must be kept at reset value. |
| 9 | WUPEN1 | WKUP pin1 (PA2) enable<br>0: Disable WKUP pin1 function<br>1: Enable WKUP pin1 function<br>If WUPEN1 is set before entering the power saving mode, a rising edge on the |

| | | |
|---|---|---|
| | | WKUP pin1 wakes up the system from the power saving mode. As the WKUP pin1 is active high, the WKUP pin1 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 8 | WUPEN0 | WKUP pin0 (PA0) enable<br>0: Disable WKUP pin0 function<br>1: Enable WKUP pin0 function<br>If WUPEN0 is set before entering the power saving mode, a rising edge on the WKUP pin0 wakes up the system from the power saving mode. As the WKUP pin0 is active high, the WKUP pin0 is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | LVDF | Low voltage detector status flag<br>0: Low Voltage event has not occurred ($V_{DD}$ is higher than the specified LVD threshold)<br>1: Low Voltage event occurred ($V_{DD}$ is equal to or lower than the specified LVD threshold)<br>**Note**: The LVD function is stopped in Standby mode. |
| 1 | STBF | Standby flag<br>0: The device has not entered the standby mode<br>1: The device has been in the standby mode<br>This bit is cleared only by a POR / PDR or by setting the STBRST bit in the PMU_CTL0 register. |
| 0 | WUF | Wakeup Flag<br>0: No wakeup event has been received<br>1: Wakeup event occurred from the WKUP pins or the RTC alarm event<br>This bit is cleared only by a POR / PDR or by setting the WURST bit in the PMU_CTL0 register. |

### 5.4.3. Control register 1 (PMU_CTL1)

Address offset: 0x08

Reset value: 0x0000 0000 (reset by wakeup from Standby mode)

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TEMPHF | TEMPLF | VBATHF | VBATLF | Reserved | | | BKPVSRF |
| | | | | | | | | r | r | r | r | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | VBTMEN | Reserved | | | BKPVSEN |
| | | | | | | | | | | | rw | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | TEMPHF | Temperature level monitoring versus high threshold<br>0: Temperature below high threshold level.<br>1: Temperature equal or above high threshold level. |
| 22 | TEMPLF | Temperature level monitoring versus low threshold<br>0: Temperature above low threshold level.<br>1: Temperature equal or below low threshold level. |
| 21 | VBATHF | VBAT level monitoring versus high threshold<br>0: VBAT level below high threshold level.<br>1: VBAT level equal or above high threshold level. |
| 20 | VBATLF | VBAT level monitoring versus low threshold<br>0: VBAT level above low threshold level.<br>1: VBAT level equal or below low threshold level. |
| 19:17 | Reserved | Must be kept at reset value. |
| 16 | BKPVSRF | Backup voltage stabilizer ready flag<br>This bit is set by hardware to indicate that the backup regulator is ready.<br>0: Backup voltage stabilizer not ready.<br>1: Backup voltage stabilizer ready. |
| 15:5 | Reserved | Must be kept at reset value. |
| 4 | VBTMEN | VBAT and temperature monitoring enable.<br>When set, the VBAT supply and temperature monitoring is enabled.<br>0: VBAT and temperature monitoring disabled.<br>1: VBAT and temperature monitoring enabled.<br>**Note**: VBAT and temperature monitoring are only available when the Backup voltage stabilizer ready (BKPVSRF bit set to 1). |
| 3:1 | Reserved | Must be kept at reset value. |
| 0 | BKPVSEN | Backup voltage stabilizer enable<br>This bit is set and cleared by software.<br>When set, the backup voltage stabilizer (used to maintain the backup RAM content in Standby and $V_{BAT}$ modes) is enabled.<br>When reset, the backup voltage stabilizer is off. The backup RAM can still be used in Run and Deep-sleep modes. However, its content will be lost in Standby and $V_{BAT}$ modes.<br>If BKPVSEN is set, the application must wait till the backup voltage stabilizer ready flag (BKPVSRF) is set. Which indicate that the data written into the SRAM will be maintained in Standby and $V_{BAT}$ modes.<br>0: Backup voltage stabilizer disabled. |

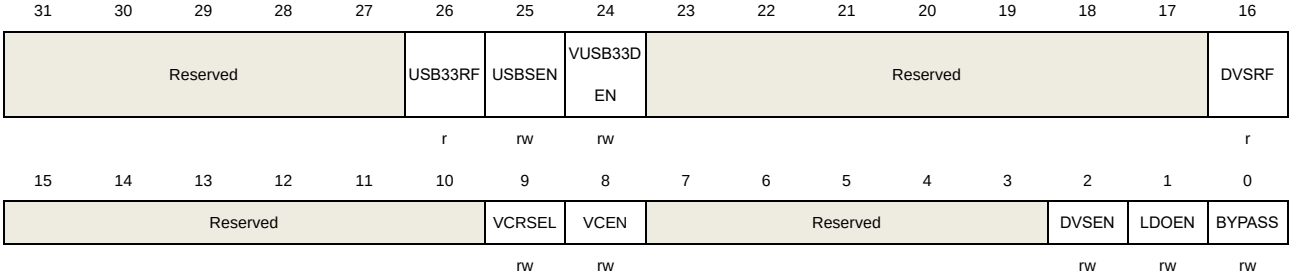
1: Backup voltage stabilizer enabled.

### 5.4.4. Control register 2 (PMU_CTL2)

Address offset: 0x10

Reset value: 0x0000 0046 (reset by wakeup from Standby mode)

This register can be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | USB33RF | USB supply ready flag bit<br>0: USB33 supply not ready.<br>1: USB33 supply ready. |
| 25 | USBSEN | USB voltage stabilizer enable.<br>This bit is set and cleared by software.<br>0: USB voltage stabilizer disabled.<br>1: USB voltage stabilizer enabled. |
| 24 | VUSB33DEN | $V_{DD33USB}$ voltage level detector enable bit<br>This bit is set and cleared by software.<br>0: $V_{DD33USB}$ voltage level detector disabled.<br>1: $V_{DD33USB}$ voltage level detector enabled. |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | DVSRF | Step-down voltage stabilizer ready flag bit<br>This bit is set by hardware to indicate that the external supply from the step-down voltage stabilizer is ready.<br>0: External supply not ready<br>1: External supply ready. |
| 15:10 | Reserved | Must be kept at reset value. |
| 9 | VCRSEL | $V_{BAT}$ battery charging resistor selection<br>0: 5kOhms resistor is selected for charing $V_{BAT}$ battery.<br>1: 1.5kOhms resistor is selected for charing $V_{BAT}$ battery. |
| 8 | VCEN | $V_{BAT}$ battery charging enable<br>0: Disable $V_{BAT}$ battery charging |

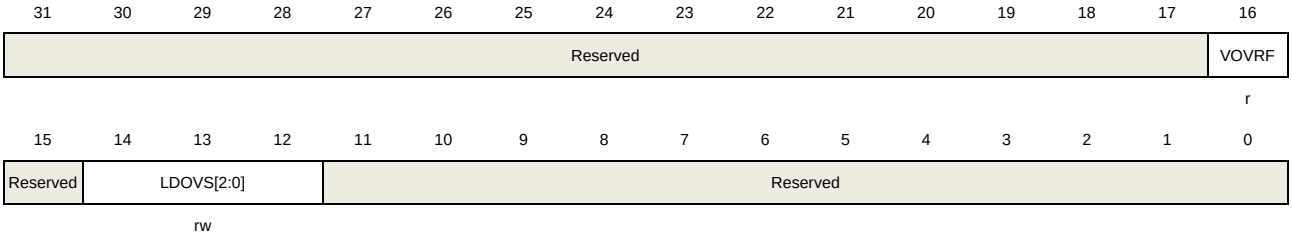
| | | |
|---|---|---|
| | | 1: Enable $V_{BAT}$ battery charging |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | DVSEN | Step-down voltage stabilizer enable bit<br>This bit is set and cleared by software.<br>0: Step-down voltage stabilizer disabled<br>1: Step-down voltage stabilizer enabled. |
| 1 | LDOEN | Low drop-out voltage stabilizer enable bit<br>This bit is set and cleared by software.<br>0: Low drop-out voltage stabilizer disabled.<br>1: Low drop-out voltage stabilizer enabled |
| 0 | BYPASS | Power management unit bypass control bit<br>This bit is set and cleared by software.<br>0: Power management unit normal operation.<br>1: Power management unit bypassed, voltage monitoring still active. |

### 5.4.5. Control register 3 (PMU_CTL3)

Address offset: 0x14

Reset value: 0x0000 2000

This register can be accessed by word (32-bit).

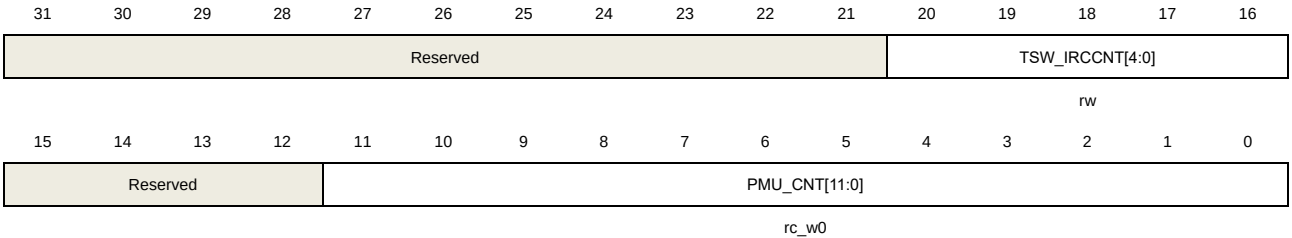


| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | VOVRF | $V_{CORE}$ voltage ready bit<br>This bit is set by hardware to indicate that the $V_{CORE}$ supply is ready.<br>0: $V_{CORE}$ supply not ready<br>1: $V_{CORE}$ supply ready. |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | LDOVS[2:0] | LDO output voltage select<br>These bits control the $V_{CORE}$ voltage level, different voltage levels and system clock frequencies will make MCU have different performance. When the performance is expected to be reduced, the system clock frequency should be reduced first, and then the LDO output voltage value should be changed. On the contrary, when the performance is improved, the LDO output voltage value should be changed first, and then the system clock frequency should be improved<br>000: LDO output 0.8V mode<br>001: LDO output 0.85V mode<br>010: LDO output 0.9V mode (default)<br>011: LDO output 0.95V mode<br>100: LDO output 0.975V mode<br>101: LDO output 1V mode<br>Other: Reserved |
| 11:0 | Reserved | Must be kept at reset value. |

### 5.4.6. Parameter register (PMU_PAR)

Address offset: 0x18
Reset value: 0x000A 0000

This register can be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | TSW_IRCCNT[4:0] | When enter Deep-sleep, switch to LPIRC4M / IRC64M (confirmed by DSPWUSSEL) clock. Wait the LPIRC4M / IRC64M (confirmed by DSPWUSSEL) counter and then set Deep-sleep mode. The default is 10 clocks. |

| | | |
|---|---|---|
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | PMU_CNT[11:0] | Exit Deep-sleep mode wait time count configure bits<br>When exit Deep-sleep mode, before open system clock, it recommend 5us~50us delay. |

# 6. Reset and clock unit (RCU)

## 6.1. Reset control unit (RCTL)

### 6.1.1. Overview

GD32H73x_75x reset control unit includes the control of three kinds of reset: power reset, system reset and backup domain reset. The power reset, known as a cold reset, resets the full system except the backup domain. The system reset resets the processor core and peripheral IP components except for the SW-DP controller and the backup domain. The backup domain reset resets the backup domain. The resets can be triggered by an external signal, internal events and the reset generators. More information about these resets will be described in the following sections.

### 6.1.2. Function overview

#### Power reset

The power reset is generated by either an external reset as power on and power down reset (POR / PDR reset), brownout reset (BOR reset) or by the internal reset generator when exiting Standby mode. The power reset sets all registers to their reset values except the Backup domain. The Power reset whose active signal is low, it will be de-asserted when the internal LDO voltage regulator is ready to provide 0.9V power. The reset service routine vector is fixed at address 0x0000_0004 in the memory map.

#### System reset

A system reset is generated by the following events:

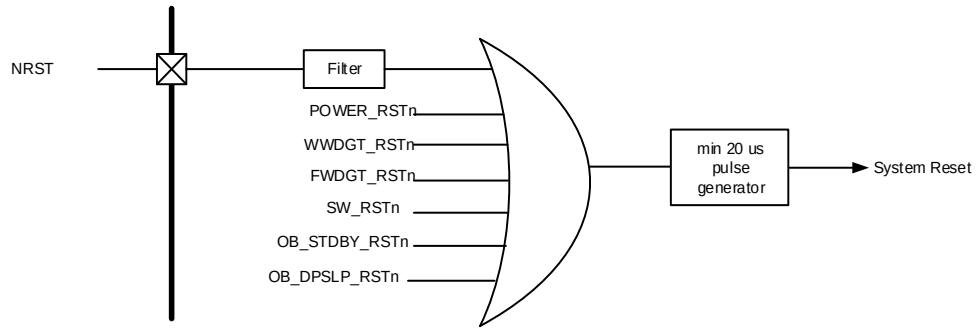
- A power reset (POWER_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT_RSTn).
- A free watchdog timer reset (FWDGT_RSTn).
- The SYSRESETREQ bit in Cortex®-M7 application interrupt and reset control register is set (SW_RSTn).
- Reset generated when entering Standby mode when resetting nRST_STDBY bit in user option bytes (OB_STDBY_RSTn).
- Reset generated when entering Deep-sleep mode when resetting nRST_DPSLP bit in user option bytes (OB_DPSLP_RSTn).

A system reset resets the processor core and peripheral IP components except for the SW-DP controller and the backup domain.

A system reset pulse generator guarantees low level pulse duration of 20 μs for each reset

source (external or internal reset).

**Figure 6-1. The system reset circuit**




### Backup domain reset

A backup domain reset is generated by setting the BKPRST bit in the Backup domain control register or backup domain power on reset (VDD or VBAT power on, if both supplies have previously been powered off).

**Note:** The BKPSRAM is not reset by backup domain reset.

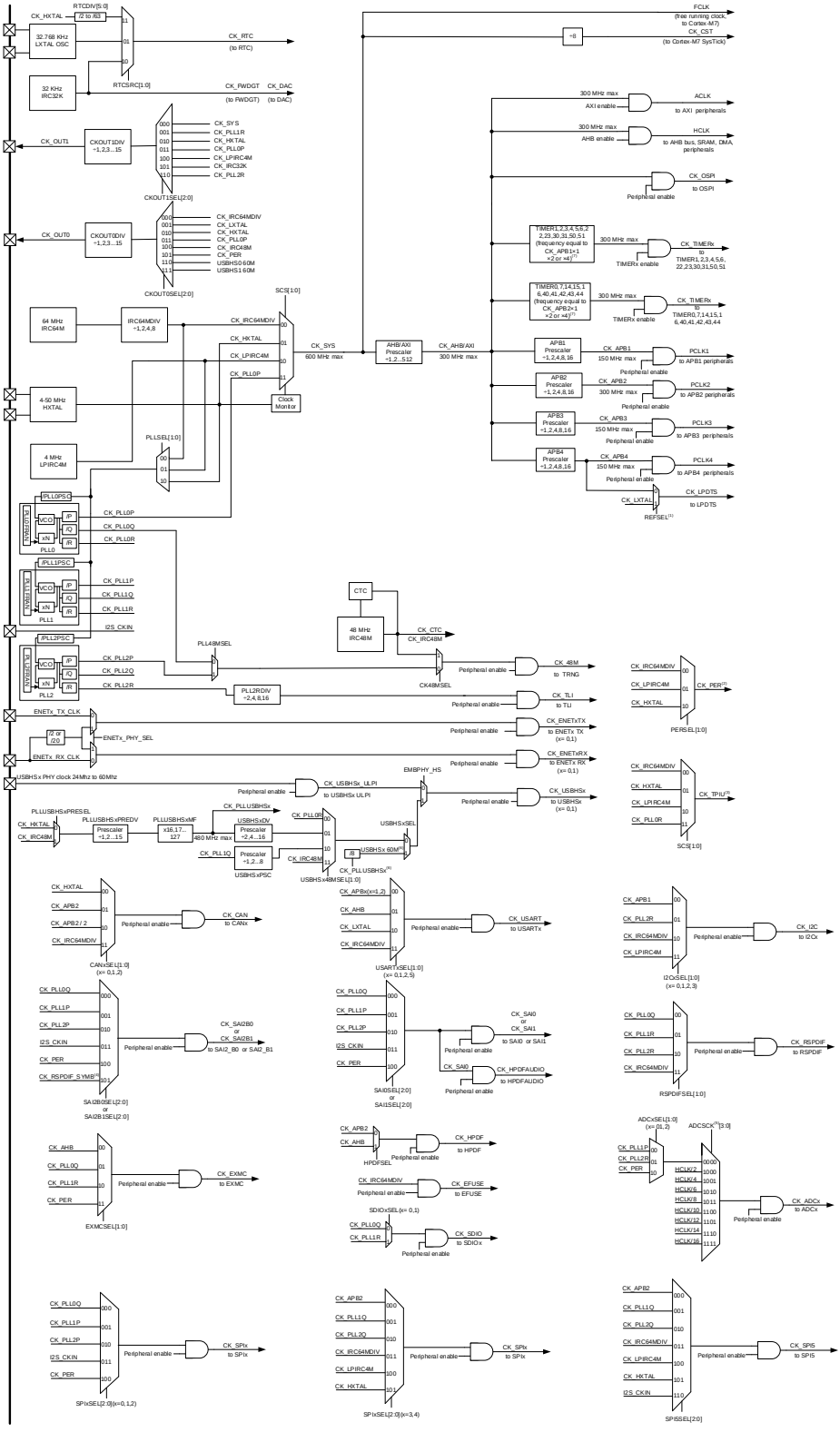
## 6.2. Clock control unit (CCTL)

### 6.2.1. Overview(For GD32H737_757_759)

The clock control unit provides a range of frequencies and clock functions. These include a Internal 64M RC oscillator (IRC64M), a Internal 48M RC oscillator (IRC48M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 32K RC oscillator (IRC32K), a Low Speed crystal oscillator (LXTAL), a Low Power Internal 4M RC oscillator (LPIRC4M), five Phase Lock Loop (PLL), a HXTAL clock monitor, a LXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AXI, AHB, APB and Cortex®-M7 are derived from the system clock (CK_SYS) which can source from the IRC64M, HXTAL, LPIRC4M or PLL0. The maximum operating frequency of the system clock (CK_SYS) can be up to 600 MHz.

**Figure 6-2. Clock tree**




(1). The REFSEL is LPDTS reference clock which is clocked by CK_APB4 or CK_LXTAL, refer to ***Figure 40-1. LPDTS block diagram***.

(2). The CK_PER is peripheral clock which is clocked by CK_IRC64MDIV, CK_LPIRC4M or CK_HXTAL.

(3). The CK_TPIU is Trace Port Interface Unit (TPIU) clock which is clocked by CK_IRC64MDIV, CK_HXTAL, CK_LPIRC4M or CK_PLL0R.

(4). The CK_RSPDIF_SYMB is RSPDIF channel symbol clock, refer to ***Figure 33-1. RSPDIF block diagram***.

(5). The ADCSCK is ADC synchronization clock seclection bits, refer to ***Sync control register (ADC_SYNCCTL)***.

(6). USBHSx 60M is USBHSx internal PHY 60M input source clock, refer to ***Internal embedded PHY***. When USBHSx 60M is selected as the USBHSx clock,CK_PLLUSBHSx should be configured at 480Mhz.

(7). TIMER clock frequency, refer to the TIMERSEL bit in the RCU_CFG1 register.

The frequency of AXI, AHB, APB4, APB3, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AXI / AHB and the APB4 / APB3 / APB2 / APB1 domains is 300 MHz / 300 MHz / 150 MHz / 150 MHz / 300 MHz / 150 MHz. The Cortex System Timer (SysTick) external clock is clocked with the system clock (CK_SYS) divided by 8. The SysTick can work either with this clock or with the system clock (CK_SYS), configurable in the SysTick control and status register.

The ADCs are clocked by the clock of PLL1P, PLL2R, CK_PER selected by the ADCxSEL or by the clock of AHB divided by 2, 4, 6, 8, 10, 12, 14, 16.

The TIMERs are clocked by the clock divided from CK_AHB. The frequency of TIMERs clock is equal to CK_APBx, twice the CK_APBx or four times the CK_APBx. Please refer to TIMERSEL bit in RCU_CFG1 for detail.

The TRNG are clocked by the clock of CK_48M. The CK_48M is selected from the clock of PLL0Q, the clock of PLL2P or the clock of IRC48M by PLL48MSEL and CK48MSEL bit in RCU_ADDCTL0 register. The TRNG supports clock switch dynamically.

The USBHS ULPI is clocked by external ULPI PHY clock or the internal clock , which select by EMBPHY_HS in USBHS_GUSBCS register. The USBHS internal clock is configured by the USBHSxSEL bits in RCU_USBCLKCTL register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

The USART is clocked by the clock of CK_APBx, CK_AHB, CK_LXTAL or CK_IRC64MDIV which defined by USARTxSEL bits in RCU_CFG1 register. The USART supports clock switch dynamically.

The I2C is clocked by the clock of CK_APB1, CK_PLL2R, CK_IRC64MDIV or CK_LPIRC4M which defined by I2CxSEL bits in RCU_CFG0 register or RCU_CFG3 register. The I2C supports clock switch dynamically.

The SPI0(I2S0), SPI1(I2S1) and SPI2(I2S2) are clocked by the clock of CK_PLL0Q, CK_PLL1P, CK_PLL2P, I2S_CKIN or CK_PER which defined by SPIxSEL(x = 0,1,2) bits in RCU_CFG5 register. The SPI0(I2S0), SPI1(I2S1) and SPI2(I2S2) support clock switch dynamically.

The SPI3 and SPI4 are clocked by the clock of CK_APB2, CK_PLL1Q, CK_PLL2Q, CK_IRC64MDIV, CK_LPIRC4M or CK_HXTAL which defined by SPIxSEL(x = 3,4) bits in RCU_CFG5 register. The SPI3 / SPI4 supports clock switch dynamically.

The SPI5(I2S5) is clocked by the clock of CK_APB2, CK_PLL1Q, CK_PLL2Q, CK_IRC64MDIV, CK_LPIRC4M, CK_HXTAL or I2S_CKIN which defined by SPI5SEL bits in

RCU_CFG5 register. The SPI5 / I2S5 supports clock switch dynamically.

The OSPI is clocked by the clock of CK_AHB.

The LPDTS is clocked by the clock of CK_APB4 or CK_LXTAL.

The CAN is clocked by the clock of CK_HXTAL, CK_APB2, CK_APB2 / 2 or CK_IRC64MDIV which defined by CANxSEL bits in RCU_CFG1 register. The CAN supports clock switch dynamically.

The RSPDIF is clocked by the clock of CK_PLL0Q, CK_PLL1R, CK_PLL2R or CK_IRC64MDIV which defined by RSPDIFSEL bits in RCU_CFG1 register. The RSPDIF supports clock switch dynamically.

The SAI2 is clocked by the clock of CK_PLL0Q, CK_PLL1P, CK_PLL2P, I2S_CKIN, CK_PER or CK_RSPDIF_SYMB which defined by SAI2B0SEL bits or SAIB1SEL bits in RCU_CFG2 register. The SAI2 supports clock switch dynamically.

The SAI0 and SAI1 are clocked by the clock of CK_PLL0Q, CK_PLL1P, CK_PLL2P, I2S_CKIN or CK_PER which defined by SAI0SEL bits or SAI1SEL bits in RCU_CFG2 register. The SAI0 and SAI1 supports clock switch dynamically.

The HPDF is clocked by the clock of CK_AHB or CK_APB2 which defined by HPDFSEL bit in RCU_CFG1 register. The HPDF supports clock switch dynamically.

The HPDF_AUDIO is clocked by the clock of CK_PLL0Q, CK_PLL1P, CK_PLL2P, I2S_CKIN or CK_PER which defined by SAI0SEL bits in RCU_CFG2 register.

The EXMC is clocked by the clock of CK_AHB, CK_PLL0Q, CK_PLL1R or CK_PER which defined by EXMCSEL bits in RCU_CFG4 register. The EXMC supports clock switch dynamically.

The SDIO is clocked by the clock of CK_PLL0Q or CK_PLL1R which defined by SDIOxSEL bits in RCU_CFG3 or RCU_CFG4 register. The SDIO supports clock switch dynamically.

The EFUSE is clocked by CK_IRC64MDIV.

The TLI is clocked by the clock of PLL2R divided by 2, 4, 8, 16 which defined by PLL2RDIV bits in RCU_CFG1 register.

The ENETx TX / RX are clocked by External PIN (ENETx_TX_CLK / ENETx_RX_CLK), which select by ENET0_PHY_SEL bit or ENET1_PHY_SEL bit in SYSCFG_PMCFG register.

The RTC is clocked by LXTAL clock or IRC32K clock or HXTAL clock divided by 2 to 63 (defined by RTCDIV bits in RCU_CFG0) which select by RTCSRC bit in backup domain control register (RCU_BDCTL). After the RTC select HXTAL clock divided by 2 to 63 (defined by RTCDIV bits in RCU_CFG0), the clock disappeared when the 0.9V core domain power off. After the RTC select IRC32K, the clock disappeared when $V_{DD}$ power off. When the RTC select LXTAL, the clock disappeared when $V_{DD}$ and $V_{BAT}$ power off.

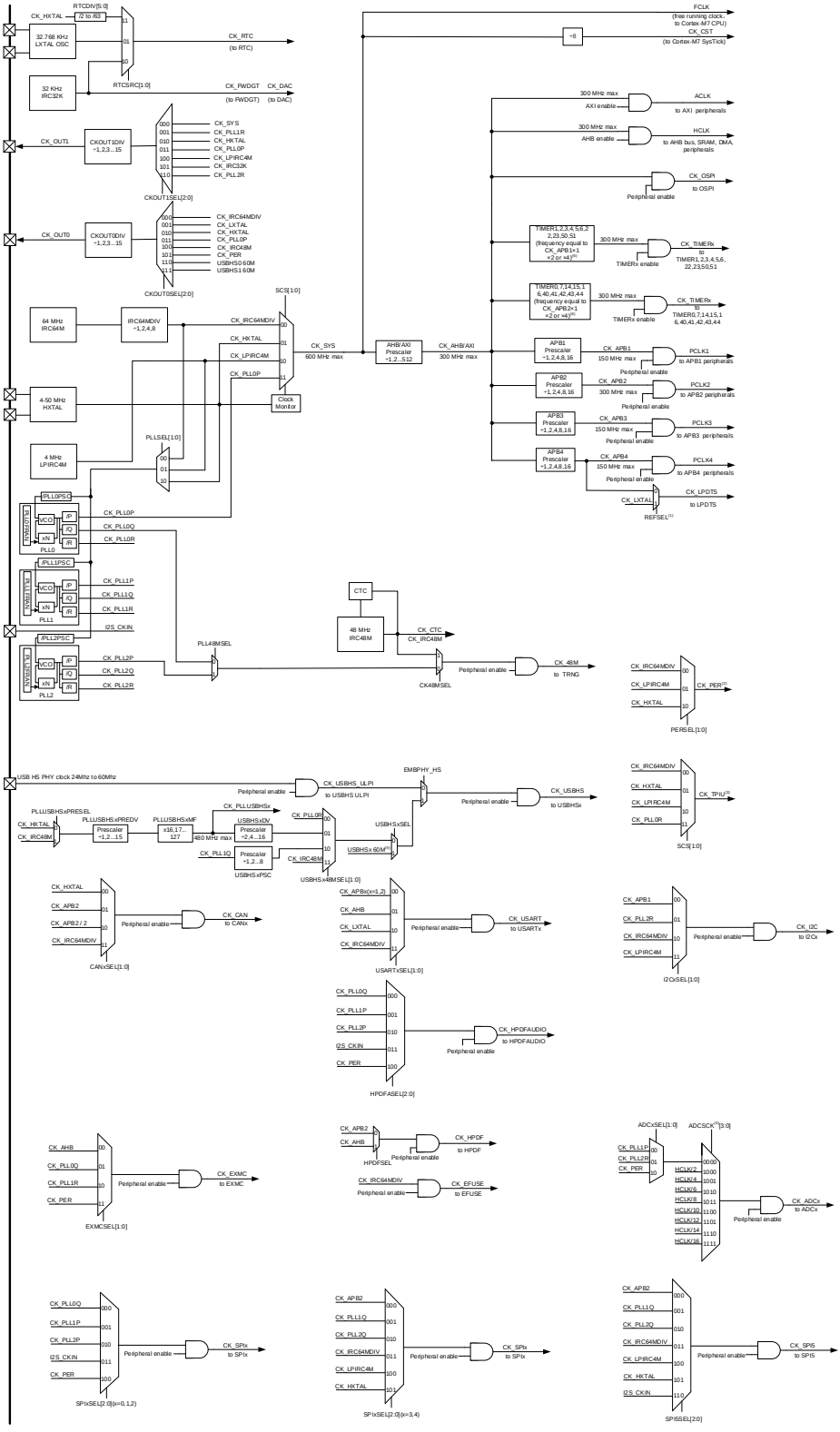
The FWDGT is clocked by IRC32K clock, which is forced on when FWDGT started.

### 6.2.2. Overview(For GD32H75E)

The clock control unit provides a range of frequencies and clock functions. These include a Internal 64M RC oscillator (IRC64M), a Internal 48M RC oscillator (IRC48M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 32K RC oscillator (IRC32K), a Low Speed crystal oscillator (LXTAL), a Low Power Internal 4M RC oscillator (LPIRC4M), five Phase Lock Loop (PLL), a HXTAL clock monitor, a LXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AXI, AHB, APB and Cortex®-M7 are derived from the system clock (CK_SYS) which can source from the IRC64M, HXTAL, LPIRC4M or PLL0. The maximum operating frequency of the system clock (CK_SYS) can be up to 600 MHz.

**Figure 6-3. Clock tree**




(1). The REFSEL is LPDTS reference clock which is clocked by CK_APB4 or CK_LXTAL, refer to ***Figure 40-1. LPDTS block diagram***.

(2). The CK_PER is peripheral clock which is clocked by CK_IRC64MDIV, CK_LPIRC4M or CK_HXTAL.

(3). The CK_TPIU is Trace Port Interface Unit (TPIU) clock which is clocked by CK_IRC64MDIV, CK_HXTAL, CK_LPIRC4M or CK_PLL0R.

(4). The ADCSCK is ADC synchronization clock seclection bits, refer to ***Sync control register (ADC_SYNCCTL)***.

(5). USBHSx 60M is USBHSx internal PHY 60M input source clock, refer to ***Internal embedded PHY***.

(6). TIMER clock frequency, refer to the TIMERSEL bit in the RCU_CFG1 register.

The frequency of AXI, AHB, APB4, APB3, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AXI / AHB and the APB4 / APB3 / APB2 / APB1 domains is 300 MHz / 300 MHz / 150 MHz / 150 MHz / 300 MHz / 150 MHz. The Cortex System Timer (SysTick) external clock is clocked with the system clock (CK_SYS) divided by 2. The SysTick can work either with this clock or with the system clock (CK_SYS), configurable in the SysTick control and status register.

The ADCs are clocked by the clock of PLL1P, PLL2R, CK_PER selected by the ADCxSEL or by the clock of AHB divided by 2, 4, 6, 8, 10, 12, 14, 16.

The TIMERs are clocked by the clock divided from CK_AHB. The frequency of TIMERs clock is equal to CK_APBx, twice the CK_APBx or four times the CK_APBx. Please refer to TIMERSEL bit in RCU_CFG1 for detail.

The TRNG are clocked by the clock of CK_48M. The CK_48M is selected from the clock of PLL0Q, the clock of PLL2P or the clock of IRC48M by PLL48MSEL and CK48MSEL bit in RCU_ADDCTL0 register. The TRNG supports clock switch dynamically.

The USBHS ULPI is clocked by external ULPI PHY clock or the internal clock , which select by EMBPHY_HS in USBHS_GUSBCS register. The USBHS internal clock is configured by the USBHSxSEL bits in RCU_USBCLKCTL register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

The USART is clocked by the clock of CK_APBx, CK_AHB, CK_LXTAL or CK_IRC64MDIV which defined by USARTxSEL bits in RCU_CFG1 register. The USART supports clock switch dynamically.

The I2C is clocked by the clock of CK_APB1, CK_PLL2R, CK_IRC64MDIV or CK_LPIRC4M which defined by I2CxSEL bits in RCU_CFG0 register or RCU_CFG3 register. The I2C supports clock switch dynamically.

The SPI0(I2S0), SPI1(I2S1) and SPI2(I2S2) are clocked by the clock of CK_PLL0Q, CK_PLL1P, CK_PLL2P, I2S_CKIN or CK_PER which defined by SPIxSEL(x = 0,1,2) bits in RCU_CFG5 register. The SPI0(I2S0), SPI1(I2S1) and SPI2(I2S2) support clock switch dynamically.

The SPI3 and SPI4 are clocked by the clock of CK_APB2, CK_PLL1Q, CK_PLL2Q, CK_IRC64MDIV, CK_LPIRC4M or CK_HXTAL which defined by SPIxSEL(x = 3,4) bits in RCU_CFG5 register. The SPI3 / SPI4 supports clock switch dynamically.

The SPI5(I2S5) is clocked by the clock of CK_APB2, CK_PLL1Q, CK_PLL2Q, CK_IRC64MDIV, CK_LPIRC4M, CK_HXTAL or I2S_CKIN which defined by SPI5SEL bits in RCU_CFG5 register. The SPI5 / I2S5 supports clock switch dynamically.

The OSPI is clocked by the clock of CK_AHB.

The LPDTS is clocked by the clock of CK_APB4 or CK_LXTAL.

The CAN is clocked by the clock of CK_HXTAL, CK_APB2, CK_APB2 / 2 or CK_IRC64MDIV which defined by CANxSEL bits in RCU_CFG1 register. The CAN supports clock switch dynamically.

The HPDF is clocked by the clock of CK_AHB or CK_APB2 which defined by HPDFSEL bit in RCU_CFG1 register. The HPDF supports clock switch dynamically.

The HPDF_AUDIO is clocked by the clock of CK_PLL0Q, CK_PLL1P, CK_PLL2P, I2S_CKIN or CK_PER which defined by HPDFASEL bits in RCU_CFG2 register.

The EXMC is clocked by the clock of CK_AHB, CK_PLL0Q, CK_PLL1R or CK_PER which defined by EXMCSEL bits in RCU_CFG4 register. The EXMC supports clock switch dynamically.

The EFUSE is clocked by CK_IRC64MDIV.

The RTC is clocked by LXTAL clock or IRC32K clock or HXTAL clock divided by 2 to 63 (defined by RTCDIV bits in RCU_CFG0) which select by RTCSRC bit in backup domain control register (RCU_BDCTL). After the RTC select HXTAL clock divided by 2 to 63 (defined by RTCDIV bits in RCU_CFG0), the clock disappeared when the 0.9V core domain power off. After the RTC select IRC32K, the clock disappeared when $V_{DD}$ power off. When the RTC select LXTAL, the clock disappeared when $V_{DD}$ and $V_{BAT}$ power off.

The FWDGT is clocked by IRC32K clock, which is forced on when FWDGT started.

### 6.2.3. Characteristics

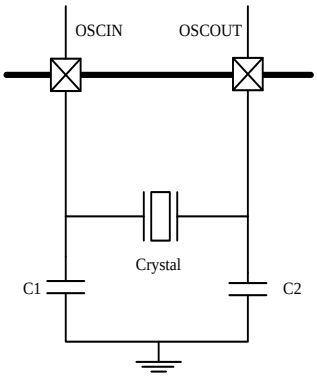
- 4 to 50 MHz High Speed crystal oscillator (HXTAL).
- Internal 64 MHz RC oscillator (IRC64M).
- Internal 48 MHz RC oscillator (IRC48M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 32KHz RC oscillator (IRC32K).
- Low Power Internal 4M RC oscillator (LPIRC4M).
- PLL clock source can be HXTAL, LPIRC4M or IRC64M.
- Integer and fraction factors of PLLs.
- PLLs fraction factors can be modified at runtime.
- Peripheral clock switch dynamically.
- HXTAL clock monitor.
- LXTAL clock monitor.

## 6.2.4. Function overview

### High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), which has a frequency from 4 to 50 MHz, produces a highly accurate clock source for use as the system clock. A crystal with a specific frequency must be connected and located close to the two HXTAL pins. The external resistor and capacitor components connected to the crystal are necessary for proper oscillation.

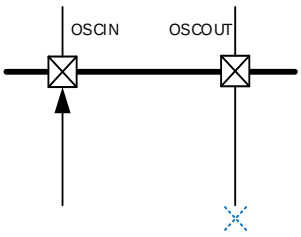
**Figure 6-4. HXTAL clock source**




The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the control register RCU_CTL. The HXTALSTB flag in control register RCU_CTL indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be released for use until this HXTALSTB bit is set by the hardware. This specific delay period is known as the oscillator “Start-up time”. As the HXTAL becomes stable, an interrupt will be generated if the related interrupt enable bit HXTALSTBIE in the Interrupt Register RCU_INT is set. At this point the HXTAL clock can be used directly as the system clock source or the PLL input clock.

Select external clock bypass mode by setting the HXTALBPS and HXTALEN bits in the control register RCU_CTL. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.During bypass mode, the signal is connected to OSCIN, and OSCOUT remains in the suspended state, as shown in ***Figure 6-5. HXTAL clock source in bypass mode***. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

**Figure 6-5. HXTAL clock source in bypass mode**



### Internal 64M RC oscillators (IRC64M)

The internal 64M RC oscillator, IRC64M, has a fixed frequency of 64 MHz and is the default clock source selection for the CPU when the device is powered up. If the IRC64MDIV[1:0] bits in RCU_ADDCTL1 register is configured, the CK_IRC64MDIV can output 8, 16, 32, 64MHz clock. The IRC64M oscillator provides a lower cost type clock source as no external components are required. The IRC64M RC oscillator can be switched on or off using the IRC64MEN bit in the control register RCU_CTL. The IRC64MSTB flag in the control register RCU_CTL is used to indicate if the internal 64M RC oscillator is stable. The start-up time of the IRC64M oscillator is shorter than the HXTAL crystal oscillator. An interrupt can be generated if the related interrupt enable bit, IRC64MSTBIE, in the Clock Interrupt Register, RCU_INT, is set when the IRC64M becomes stable. The CK_IRC64MDIV clock can also be used as the system clock source or the PLL input clock.

The frequency accuracy of the IRC64M can be calibrated by the manufacturer, but its operating frequency is still less accurate than HXTAL. The application requirements, environment and cost will determine which oscillator type is selected.

If the HXTAL or PLL0P is the system clock source, to minimize the time required for the system to recover from the Deep-sleep Mode, the hardware forces the CK_IRC64MDIV or CK_LPIR4M clock to be the system or kernel clock when the system initially wakes-up.

### Internal 48M RC oscillators (IRC48M)

The internal 48M RC oscillator, IRC48M, has a fixed frequency of 48 MHz. The IRC48M oscillator provides a lower cost type clock source as no external components are required when USBHS / TRNG used. The IRC48M RC oscillator can be switched on or off using the IRC48MEN bit in the RCU_ADDCTL0 Register. The IRC48MSTB flag in the RCU_ADDCTL0 Register is used to indicate if the internal 48M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC48MSTBIE, in the RCU_ADDINT Register, is set when the IRC48M becomes stable. The IRC48M clock is used for the clocks of USBHS / TRNG.

The frequency accuracy of the IRC48M can be calibrated by the manufacturer, but its operating frequency is still not enough accurate because the USBHS need the frequency must between 48MHz±1%. A hardware automatically dynamic trim performed in CTC unit adjust the IRC48M to the needed frequency.

### Low speed crystal oscillator (LXTAL)

The low speed external crystal or ceramic resonator oscillator, which has a frequency of 32,768 Hz, produces a low power but highly accurate clock source for the Real Time Clock circuit. The LXTAL oscillator can be switched on or off using the LXTALEN bit in the backup domain control register (RCU_BDCTL). The LXTALSTB flag in the backup domain control register (RCU_BDCTL) will indicate if the LXTAL clock is stable. An interrupt can be generated if the related interrupt enable bit, LXTALSTBIE, in the Interrupt Register RCU_INT is set when the LXTAL becomes stable.

Select external clock bypass mode by setting the LXTALBPS and LXTALEN bits in the backup domain control register (RCU_BDCTL). The CK_LXTAL is equal to the external clock which drives the OSC32IN pin.

### Internal 32K RC oscillator (IRC32K)

The internal RC oscillator has a frequency of about 32 kHz and is a low power clock source for the real time clock circuit or the rree watchdog timer. The IRC32K offers a low cost clock source as no external components are required. The IRC32K RC oscillator can be switched on or off by using the IRC32KEN bit in the Reset source / clock register (RCU_RSTSCK). The IRC32KSTB flag in the reset source / clock register RCU_RSTSCK will indicate if the IRC32K clock is stable. An interrupt can be generated if the related interrupt enable bit IRC32KSTBIE in the clock interrupt register (RCU_INT) is set when the IRC32K becomes stable.

### Low Power Internal 4M RC oscillator (LPIRC4M)

The low power internal 4M RC oscillator has a frequency of about 4 MHz and is a low power clock source for use as the system clock or PLL input clock. The LPIRC4M offers a low cost clock source as no external components are required. The LPIRC4M RC oscillator can be switched on or off by using the LPIRC4MEN bit in the additional clock control register 1 (RCU_ADDCTL1). The LPIRC4MSTB flag in the clock interrupt register (RCU_INT) will indicate if the LPIRC4M clock is stable. An interrupt can be generated if the related interrupt enable bit LPIRC4MSTBIE in the clock interrupt register (RCU_INT) is set when the LPIRC4M becomes stable.

The frequency accuracy of the LPIRC4M can be calibrated by the manufacturer. After reset, the calibration is loaded in LPIRC4MCALIB bits in RCU_ADDCTL1 register.

If the HXTAL or PLL0P is the system clock source, to minimize the time required for the system to recover from the Deepsleep Mode, the hardware forces the CK_IRC64MDIV or CK_LPIR4M clock to be the system or kernel clock when the system initially wakes-up.

### Phase locked loop (PLL)

There are five internal Phase Locked Loop, the PLL0, PLL1, PLL2, PLLUSBHS0 and PLLUSBHS1. The PLL0, PLL1 and PLL2 support integer and fraction factors and the fraction factors can be modified at run time. Otherwise, the PLL0, PLL1 and PLL2 can generate P / Q / R clock output respectively. The PLL0P could be used to generator system clock (no more than 600MHz).

For each PLL, When PLLxFRAEN bit is '1' and PLLxFRAN value is not '0' in RCU_PLLxFRA register, the PLLx is in fraction mode, for example:

$$CK_PLL0VCO = CK_PLL0VCOSRC * \left(PLL0N + \frac{PLL0FRAN}{2^{13}}\right) \quad (6\text{-}1)$$

Otherwise, the PLLx is in integer mode, for example:

CK_PLL0VCO= CK_PLL0VCOSRC*PLL0N (6-2)

The PLL0 can be switched on or off by using the PLL0EN bit in the RCU_CTL Register. The PLL0STB flag in the RCU_CTL register will indicate if the PLL0 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLL0STBIE, in the RCU_INT register, is set as the PLL0 becomes stable.

The PLL1 can be switched on or off by using the PLL1EN bit in the RCU_CTL register. The PLL1STB flag in the RCU_CTL register will indicate if the PLL1 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLL1STBIE, in the RCU_INT register, is set as the PLL1 becomes stable.

The PLL2 can be switched on or off by using the PLL2EN bit in the RCU_CTL register. The PLL2STB flag in the RCU_CTL register will indicate if the PLL2 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLL2STBIE, in the RCU_INT register, is set as the PLL2 becomes stable.

The PLLUSBHS0 can be switched on or off by using the PLLUSBHS0EN bit in the RCU_ADDCTL1 register. The PLLUSBHS0STB flag in the RCU_ADDCTL1 register will indicate if the PLLUSBHS0 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLUSBHS0STBIE, in the RCU_ADDINT register, is set as the PLLUSBHS0 becomes stable.

The PLLUSBHS1 can be switched on or off by using the PLLUSBHS1EN bit in the RCU_ADDCTL1 register. The PLLUSBHS1STB flag in the RCU_ADDCTL1 register will indicate if the PLLUSBHS1 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLUSBHS1STBIE, in the RCU_ADDINT register, is set as the PLLUSBHS1 becomes stable.

The five PLLs are closed by hardware when entering the Deepsleep / Standby mode or HXTAL monitor fail when HXTAL used as the source clock of the PLLs.

### Peripheral clock switch dynamically

If the peripheral has two more source clock selection, the peripheral can switches from the running clock to another present clock dynamically. Otherwise, peripheral clock can not be switched. For GD32H737_757_759, RNG / USART / I2C / SPI / RSPDIF / SAI / SDIO / EXMC / CAN / HPDF peripherals support clock switch dynamically. For GD32H75E, TRNG / USART / I2C / SPI / EXMC / CAN / HPDF peripherals support clock switch dynamically.

### System clock (CK_SYS) selection

After the system reset, the default CK_SYS source will be IRC64M and can be switched to HXTAL or LPIRC4M or CK_PLL0P by changing the system clock switch bits, SCS, in the clock configuration register 0, RCU_CFG0. When the SCS value is changed, the CK_SYS will continue to operate using the original clock source until the target clock source is stable. When a clock source is directly or indirectly (by PLL0P) used as the CK_SYS, it is not possible to stop it.

### HXTAL clock monitor (CKM)

The HXTAL clock monitor function is enabled by the HXTAL clock monitor enable bit, CKMEN, in the control register (RCU_CTL). This function should be enabled after the HXTAL start-up delay and disabled when the HXTAL is stopped. Once the HXTAL failure is detected, the HXTAL will be automatically disabled. The HXTAL clock stuck interrupt flag, CKMIF, in the clock interrupt register, RCU_INT, will be set and the HXTAL failure event will be generated. This failure interrupt is connected to the Non-Maskable Interrupt, NMI, of the Cortex®-M7. If the HXTAL is selected as the clock source of CK_SYS or PLL0 and CK_PLL0P used as system clock, the HXTAL failure will force the CK_SYS source to IRC64M and the PLL0 will be disabled automatically. If the HXTAL is selected as the clock source of any PLLs, the HXTAL failure will force the PLL closed automatically.

### LXTAL clock monitor (LCKM)

A clock monitor on LXTAL can be activated by software writing the LCKMEN bit in the backup domain control register (RCU_BDCTL). LCKMEN can not be enabled before LXTAL is enabled and ready.

The clock monitor on LXTAL is working in all modes except $V_{BAT}$. If a failure is detected on the external 32 kHz oscillator, an interrupt can be sent to CPU.

The software must then disable the LCKMEN bit, stop the defective 32 kHz oscillator, and change the RTC clock source, or take any required action to secure the application.

A 4-bits plus one counter will work at IRC32K domain when LCKMD enable. If the LXTAL clock has stuck at 0/1 error or slow down about 20KHz, the counter will overflow. The LXTAL clock failure will been found. Once the LXTAL failure is detected, the LXTAL clock stuck interrupt flag, LCKMIF, in the clock interrupt register, RCU_INT, will be set and the LXTAL failure event will be generated.

### Clock output capability

The clock output capability is ranging from 32 kHz to 600 MHz. There are several clock signals can be selected via the CK_OUT0 clock source selection bits, CKOUT0SEL, in the Clock configuration register 2 (RCU_CFG2). The corresponding GPIO pin should be configured in the properly alternate function I/O (AFIO) mode to output the selected clock signal. The CK_OUT1 is seleced by CKOUT1SEL, in the clock configuration register 2 (RCU_CFG2).

**Table 6-1. Clock output 0 source select**

| Clock Source 0 Selection bits | Clock Source |
|---|---|
| 000 | CK_IRC64MDIV |
| 001 | CK_LXTAL |
| 010 | CK_HXTAL |
| 011 | CK_PLL0P |
| 100 | CK_IRC48M |
| 101 | CK_PER |

| Clock Source 0 Selection bits | Clock Source |
|---|---|
| 110 | USBHS0 60M |
| 111 | USBHS1 60M |

**Table 6-2. Clock output 1 source select**

| Clock Source 1 Selection bits | Clock Source |
|---|---|
| 000 | CK_SYS |
| 001 | CK_PLL1R |
| 010 | CK_HXTAL |
| 011 | CK_PLL0P |
| 100 | CK_LPIRC4M |
| 101 | CK_IRC32K |
| 110 | CK_PLL2R |

The CK_OUT0 frequency can be reduced by a configurable binary divider, controlled by the CKOUT0DIV bits, in the clock cnfiguration register (RCU_CFG2).

The CK_OUT1 frequency can be reduced by a configurable binary divider, controlled by the CKOUT1DIV bits, in the clock configuration register (RCU_CFG2).

### RTC clock measure

The three clock source of RTC clock, LXTAL, IRC32K, HXTAL divided by 2 to 63 (defined by RTCDIV bits in RCU_CFG0), can be measured by TIMER. Then the user can get the clocks frequency, and adjust the RTC and FWDGT counter. Please refer to TIMER15_CI0_SEL bits and TIMER16_CI0_SEL bits ***SYSCFG_TIMERCISEL6*** regiser for detail.

## 6.3. Register definition(For GD32H737_757_759)

RCU base address: 0x5802 4400

### 6.3.1. Control register (RCU_CTL)

Address offset: 0x00

Reset value: 0xC000 8040

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC64MSTB | IRC64MEN | PLL2STB | PLL2EN | PLL1STB | PLL1EN | PLL0STB | PLL0EN | Reserved | | | | CKMEN | HXTALBPS | HXTALSTB | HXTALEN |
| r | rw | r | rw | r | rw | r | rw | | | | | rw | rw | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC64MCALIB[8:0] | | | | | | | | | IRC64MADJ[6:0] | | | | | | |
| r | | | | | | | | | rw | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | IRC64MSTB | IRC64M internal 64MHz RC oscillator stabilization flag<br>Set by hardware to indicate if the IRC64M oscillator is stable and ready for use.<br>0: IRC64M oscillator is not stable<br>1: IRC64M oscillator is stable |
| 30 | IRC64MEN | Internal 64MHz RC oscillator enable<br>Set and reset by software. This bit cannot be reset if the IRC64M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when CKMEN is set.<br>0: Internal 64 MHz RC oscillator disabled<br>1: Internal 64 MHz RC oscillator enabled |
| 29 | PLL2STB | PLL2 clock stabilization flag<br>Set by hardware to indicate if the PLL2 output clock is stable and ready for use.<br>0: PLL2 is not stable<br>1: PLL2 is stable |
| 28 | PLL2EN | PLL2 enable<br>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL2 is switched off<br>1: PLL2 is switched on |
| 27 | PLL1STB | PLL1 clock stabilization flag<br>Set by hardware to indicate if the PLL1 output clock is stable and ready for use.<br>0: PLL1 is not stable<br>1: PLL1 is stable |

| | | |
|---|---|---|
| 26 | PLL1EN | PLL1 enable<br>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL1 is switched off<br>1: PLL1 is switched on |
| 25 | PLL0STB | PLL0 clock stabilization flag<br>Set by hardware to indicate if the PLL0 output clock is stable and ready for use.<br>0: PLL0 is not stable<br>1: PLL0 is stable |
| 24 | PLL0EN | PLL0 enable<br>Set and reset by software. This bit cannot be reset if the PLL0 clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL0 is switched off<br>1: PLL0 is switched on |
| 23:20 | Reserved | Must be kept at reset value. |
| 19 | CKMEN | HXTAL clock monitor enable<br>0: Disable the High speed 4 ~ 50 MHz crystal oscillator (HXTAL) clock monitor<br>1: Enable the High speed 4 ~ 50 MHz crystal oscillator (HXTAL) clock monitor<br>When the hardware detects that the HXTAL clock is stuck at a low or high state, the internal hardware will switch the system clock to be the internal high speed IRC64M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software.<br>**Note:** When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC64M internal RC oscillator regardless of the control bit, IRC64MEN, state. |
| 18 | HXTALBPS | High speed crystal oscillator (HXTAL) clock bypass mode enable<br>The HXTALBPS bit can be written only if the HXTALEN is 0.<br>0: Disable the HXTAL Bypass mode<br>1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock. |
| 17 | HXTALSTB | High speed crystal oscillator (HXTAL) clock stabilization flag<br>Set by hardware to indicate if the HXTAL oscillator is stable and ready for use.<br>0: HXTAL oscillator is not stable<br>1: HXTAL oscillator is stable |
| 16 | HXTALEN | High speed crystal oscillator (HXTAL) enable<br>Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL0 input clock when PLL0P clock is selected to the system clock. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: High speed 4 ~ 50 MHz crystal oscillator disabled<br>1: High speed 4 ~ 50 MHz crystal oscillator enabled |

| | | |
|---|---|---|
| 15:7 | IRC64MCALIB[8:0] | Internal 64MHz RC Oscillator calibration value<br>These bits are load automatically at power on. |
| 6:0 | IRC64MADJ[6:0] | Internal 64MHz RC Oscillator clock trim adjust value<br>These bits are set by software. The trimming value is these bits IRC64MADJ[6:0] added to the IRC64MCALIB[8:0] bits. The trimming value should trim the IRC64M to 64 MHz ± 1%. |

### 6.3.2. PLL0 register (RCU_PLL0)

Address offset: 0x04

Reset value: 0x0100 2020

To configure the PLL0 clock, refer to the following formula:

CK_PLL0VCOSRC = CK_PLL0SRC / PLL0PSC

CK_PLL0VCO = CK_PLL0VCOSRC × (PLL0N + PLL0FRAN / $2^{13}$)

CK_PLL0R = CK_PLL0VCO / PLL0R

CK_PLL0P = CK_PLL0VCO / PLL0P

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLLSTBSRC | PLL0R[6:0] | | | | | | | Reserved | PLL0P[6:0] | | | | | | |
| w | rw | | | | | | | | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL0N[8:0] | | | | | | | | | PLL0PSC[5:0] | | | | | |
| | rw | | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLLSTBSRC | PLLs stabilization signal sources.<br>0: Analog logic<br>1: Digital logic |
| 30:24 | PLL0R[6:0] | The PLL0R output frequency division factor from PLL0 VCO clock<br>Set and reset by software when the PLL0 is disable. These bits used to generator PLL0R output clock (CK_PLL0R) from PLL0 VCO clock (CK_PLL0VCO). The CK_PLL0VCO is described in PLL0N bits in RCU_PLL0 register.<br>0000000: CK_PLL0R = CK_PLL0VCO<br>0000001: CK_PLL0R = CK_PLL0VCO / 2<br>0000010: CK_PLL0R = CK_PLL0VCO / 3.<br>0000011: CK_PLL0R = CK_PLL0VCO / 4<br>0000100: CK_PLL0R = CK_PLL0VCO / 5<br>…<br>1111111: CK_PLL0R = CK_PLL0VCO / 128 |
| 23 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 22:16 | PLL0P[6:0] | The PLL0P output frequency division factor from PLL0 VCO clock<br>Set and reset by software when the PLL0 is disable. These bits used to generator PLL0P output clock (CK_PLL0P) from PLL0 VCO clock (CK_PLL0VCO). The CK_PLL0P is used to system clock (no more than 600MHz). The CK_PLL0VCO is described in PLL0N bits in RCU_PLL0 register.<br>0000000: CK_PLL0P = CK_PLL0VCO<br>0000001: CK_PLL0P = CK_PLL0VCO / 2<br>0000010: CK_PLL0P = CK_PLL0VCO / 3<br>0000011: CK_PLL0P = CK_PLL0VCO / 4<br>0000100: CK_PLL0P = CK_PLL0VCO / 5<br>…<br>1111111: CK_PLL0P = CK_PLL0VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:6 | PLL0N[8:0] | The PLL0 VCO clock multiplication factor<br>Set and reset by software (only use word / half-word write) when the PLL0 is disable. These bits used to generator PLL0 VCO clock (CK_PLL0VCO) from PLL0 VCO source clock (CK_PLL0VCOSRC). The CK_PLL0VCOSRC is described in PLL0PSC bits in RCU_PLL0 register.<br>**Note:** The frequency of CK_PLL0VCO is between 150MHz to 836MHz<br>The value of PLL0N must :<br>When PLL0 fractional latch is disabled, the value of PLL0N must satisfy: 9 ≤ PLL0N ≤ 512<br>When PLL0 fractional latch is enabled, the value of PLL0N must satisfy: 12 ≤ PLL0N ≤ 508<br>000000000: Reserved<br>…<br>000000111: Reserved<br>000001000: PLL0N = 9<br>…<br>001000000: PLL0N = 65<br>001000001: PLL0N = 66<br>…<br>111111111: PLL0N = 512 |
| 5:0 | PLL0PSC[5:0] | The PLL0 VCO source clock prescaler<br>Set and reset by software when the PLL0 is disable. These bits used to generate the clock of PLL0 VCO source clock (CK_PLL0VCOSRC) from PLL0 source clock (CK_PLL0SRC) which described in PLLSEL in RCU_PLLALL register. The VCO source clock is between 1M to 16MHz<br>000000: reserved<br>000001: CK_PLL0SRC<br>000010: CK_PLL0SRC / 2<br>000011: CK_PLL0SRC / 3<br>… |

111111: CK_PLL0SRC / 63

### 6.3.3. Clock configuration register 0 (RCU_CFG0)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2C0SEL[1:0] | | APB3PSC[2:0] | | | APB4PSC[2:0] | | | Reserved | | RTCDIV[5:0] | | | | | |
| rw | | rw | | | rw | | | | | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| APB2PSC[2:0] | | | APB1PSC[2:0] | | | Reserved | | AHBPSC[3:0] | | | | SCSS[1:0] | | SCS[1:0] | |
| rw | | | rw | | | | | rw | | | | r | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | I2C0SEL[1:0] | I2C0 clock source selection<br>Set and reset by software to control the I2C0 clock source.<br>00: CK_APB1 selected as I2C0 source clock<br>01: CK_PLL2R selected as I2C0 source clock<br>10: CK_IRC64MDIV selected as I2C0 source clock<br>11: CK_LPIRC4M selected as I2C0 source clock |
| 29:27 | APB3PSC[2:0] | APB3 prescaler selection<br>Set and reset by software to control the APB3 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 26:24 | APB4PSC[2:0] | APB4 prescaler selection<br>Set and reset by software to control the APB4 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 23:22 | Reserved | Must be kept at reset value. |
| 21:16 | RTCDIV[5:0] | RTC clock divider factor<br>Set and reset by software. These bits is used to generator clock for RTC (no more than 1MHz) from HXTAL clock.<br>000000: no clock for RTC<br>000001: no clock for RTC |

| | | |
|---|---|---|
| | | 000010: CK_HXTAL / 2<br>000011: CK_HXTAL / 3<br>…<br>111111: CK_HXTAL / 63 |
| 15:13 | APB2PSC[2:0] | APB2 prescaler selection<br>Set and reset by software to control the APB2 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 12:10 | APB1PSC[2:0] | APB1 prescaler selection<br>Set and reset by software to control the APB1 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:4 | AHBPSC[3:0] | AHB and AXI prescaler selection<br>Set and reset by software to control the AHB and AXI clock division ratio<br>0xxx: CK_SYS selected<br>1000: (CK_SYS / 2) selected<br>1001: (CK_SYS / 4) selected<br>1010: (CK_SYS / 8) selected<br>1011: (CK_SYS / 16) selected<br>1100: (CK_SYS / 64) selected<br>1101: (CK_SYS / 128) selected<br>1110: (CK_SYS / 256) selected<br>1111: (CK_SYS / 512) selected |
| 3:2 | SCSS[1:0] | System clock switch status<br>Set and reset by hardware to indicate the clock source of system clock.<br>00: Select CK_IRC64MDIV as the CK_SYS source<br>01: Select CK_HXTAL as the CK_SYS source<br>10: Select CK_LPIRC4M as the CK_SYS source<br>11: Select CK_PLL0P as the CK_SYS source |
| 1:0 | SCS[1:0] | System clock switch<br>Set by software to select the CK_SYS source. Because the change of CK_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC64MDIV when leaving Deep-sleep and Standby mode or HXTAL failure is detected by HXTAL clock monitor when |

HXTAL is selected directly or indirectly as the clock source of CK_SYS.

00: Select CK_IRC64MDIV as the CK_SYS source

01: Select CK_HXTAL as the CK_SYS source

10: Select CK_LPIRC4M as the CK_SYS source

11: Select CK_PLL0P as the CK_SYS source

### 6.3.4. Clock interrupt register (RCU_INT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | LCKMIC | LCKMIF | LPIRC4MSTBIC | LPIRC4MSTBIE | LPIRC4MSTBIF | CKMIC | PLL2STBIC | PLL1STBIC | PLL0STBIC | HXTALSTBIC | IRC64MSTBIC | LXTALSTBIC | IRC32KSTBIC |
| | | | w | r | w | rw | r | w | w | w | w | w | w | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL2STBIE | PLL1STBIE | PLL0STBIE | HXTALSTBIE | IRC64MSTBIE | LXTALSTBIE | IRC32KSTBIE | CKMIF | PLL2STBIF | PLL1STBIF | PLL0STBIF | HXTALSTBIF | IRC64MSTBIF | LXTALSTBIF | IRC32KSTBIF |
| | rw | rw | rw | rw | rw | rw | rw | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | LCKMIC | LXTAL clock stuck interrupt clear<br>Write 1 by software to reset the LCKMIF flag.<br>0: Not reset LCKMIF flag<br>1: Reset LCKMIF flag |
| 27 | LCKMIF | LXTAL clock stuck interrupt flag<br>Set by hardware when the LXTAL clock is stuck.<br>Reset when setting the LCKMIC bit by software.<br>0: Clock operating normally<br>1: LXTAL clock stuck |
| 26 | LPIRC4MSTBIC | LPIRC4M stabilization interrupt clear<br>Write 1 by software to reset the LPIRC4MSTBIF flag.<br>0: Not reset LPIRC4MSTBIF flag<br>1: Reset LPIRC4MSTBIF flag |
| 25 | LPIRC4MSTBIE | LPIRC4M stabilization interrupt enable<br>Set and reset by software to enable/disable the LPIRC4M stabilization interrupt.<br>0: Disable the LPIRC4M stabilization interrupt<br>1: Enable the LPIRC4M stabilization interrupt |
| 24 | LPIRC4MSTBIF | LPIRC4M Clock Stuck Interrupt Flag |

| | | |
|---|---|---|
| | | Set by hardware when the LPIRC4M clock is stuck.<br>Reset when setting the LPIRC4MSTBIC bit by software.<br>0: Clock operating normally<br>1: LPIRC4M clock stuck |
| 23 | CKMIC | HXTAL clock stuck interrupt clear<br>Write 1 by software to reset the CKMIF flag.<br>0: Not reset CKMIF flag<br>1: Reset CKMIF flag |
| 22 | PLL2STBIC | PLL2 stabilization interrupt clear<br>Write 1 by software to reset the PLL2STBIF flag.<br>0: Not reset PLL2STBIF flag<br>1: Reset PLL2STBIF flag |
| 21 | PLL1STBIC | PLL1 stabilization interrupt clear<br>Write 1 by software to reset the PLL1STBIF flag.<br>0: Not reset PLL1STBIF flag<br>1: Reset PLL1STBIF flag |
| 20 | PLL0STBIC | PLL0 stabilization interrupt clear<br>Write 1 by software to reset the PLL0STBIF flag.<br>0: Not reset PLL0STBIF flag<br>1: Reset PLL0STBIF flag |
| 19 | HXTALSTBIC | HXTAL stabilization interrupt clear<br>Write 1 by software to reset the HXTALSTBIF flag.<br>0: Not reset HXTALSTBIF flag<br>1: Reset HXTALSTBIF flag |
| 18 | IRC64MSTBIC | IRC64M stabilization interrupt clear<br>Write 1 by software to reset the IRC64MSTBIF flag.<br>0: Not reset IRC64MSTBIF flag<br>1: Reset IRC64MSTBIF flag |
| 17 | LXTALSTBIC | LXTAL stabilization interrupt clear<br>Write 1 by software to reset the LXTALSTBIF flag.<br>0: Not reset LXTALSTBIF flag<br>1: Reset LXTALSTBIF flag |
| 16 | IRC32KSTBIC | IRC32K stabilization interrupt clear<br>Write 1 by software to reset the IRC32KSTBIF flag.<br>0: Not reset IRC32KSTBIF flag<br>1: Reset IRC32KSTBIF flag |
| 15 | Reserved | Must be kept at reset value. |
| 14 | PLL2STBIE | PLL2 stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL2 stabilization interrupt. |

| | | |
|---|---|---|
| | | 0: Disable the PLL2 stabilization interrupt<br>1: Enable the PLL2 stabilization interrupt |
| 13 | PLL1STBIE | PLL1 stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL1 stabilization interrupt.<br>0: Disable the PLL1 stabilization interrupt<br>1: Enable the PLL1 stabilization interrupt |
| 12 | PLL0STBIE | PLL0 stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL0 stabilization interrupt.<br>0: Disable the PLL0 stabilization interrupt<br>1: Enable the PLL0 stabilization interrupt |
| 11 | HXTALSTBIE | HXTAL stabilization interrupt enable<br>Set and reset by software to enable/disable the HXTAL stabilization interrupt<br>0: Disable the HXTAL stabilization interrupt<br>1: Enable the HXTAL stabilization interrupt |
| 10 | IRC64MSTBIE | IRC64M stabilization interrupt enable<br>Set and reset by software to enable/disable the IRC64M stabilization interrupt<br>0: Disable the IRC64M stabilization interrupt<br>1: Enable the IRC64M stabilization interrupt |
| 9 | LXTALSTBIE | LXTAL stabilization interrupt enable<br>LXTAL stabilization interrupt enable/disable control<br>0: Disable the LXTAL stabilization interrupt<br>1: Enable the LXTAL stabilization interrupt |
| 8 | IRC32KSTBIE | IRC32K stabilization interrupt enable<br>IRC32K stabilization interrupt enable/disable control<br>0: Disable the IRC32K stabilization interrupt<br>1: Enable the IRC32K stabilization interrupt |
| 7 | CKMIF | HXTAL clock stuck interrupt flag<br>Set by hardware when the HXTAL clock is stuck.<br>Reset when setting the CKMIC bit by software.<br>0: Clock operating normally<br>1: HXTAL clock stuck |
| 6 | PLL2STBIF | PLL2 stabilization interrupt flag<br>Set by hardware when the PLL2 is stable and the PLL2STBIE bit is set.<br>Reset when setting the PLL2STBIC bit by software.<br>0: No PLL2 stabilization interrupt generated<br>1: PLL2 stabilization interrupt generated |
| 5 | PLL1STBIF | PLL1 stabilization interrupt flag<br>Set by hardware when the PLL1 is stable and the PLL1STBIE bit is set.<br>Reset when setting the PLL1STBIC bit by software.<br>0: No PLL1 stabilization interrupt generated |

| | | |
|---|---|---|
| | | 1: PLL1 stabilization interrupt generated |
| 4 | PLL0STBIF | PLL0 stabilization interrupt flag<br>Set by hardware when the PLL0 is stable and the PLL0STBIE bit is set.<br>Reset when setting the PLL0STBIC bit by software.<br>0: No PLL0 stabilization interrupt generated<br>1: PLL0 stabilization interrupt generated |
| 3 | HXTALSTBIF | HXTAL stabilization interrupt flag<br>Set by hardware when the High speed 4 ~ 50 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.<br>Reset when setting the HXTALSTBIC bit by software.<br>0: No HXTAL stabilization interrupt generated<br>1: HXTAL stabilization interrupt generated |
| 2 | IRC64MSTBIF | IRC64M stabilization interrupt flag<br>Set by hardware when the Internal 64 MHz RC oscillator clock is stable and the IRC64MSTBIE bit is set.<br>Reset when setting the IRC64MSTBIC bit by software.<br>0: No IRC64M stabilization interrupt generated<br>1: IRC64M stabilization interrupt generated |
| 1 | LXTALSTBIF | LXTAL stabilization interrupt flag<br>Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set.<br>Reset when setting the LXTALSTBIC bit by software.<br>0: No LXTAL stabilization interrupt generated<br>1: LXTAL stabilization interrupt generated |
| 0 | IRC32KSTBIF | IRC32K stabilization interrupt flag<br>Set by hardware when the Internal 32kHz RC oscillator clock is stable and the IRC32KSTBIE bit is set.<br>Reset when setting the IRC32KSTBIC bit by software.<br>0: No IRC32K stabilization clock ready interrupt generated<br>1: IRC32K stabilization interrupt generated |

### 6.3.5. AHB1 reset register (RCU_AHB1RST)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | USBHS1RST | Reserved | | | ENET0RST | Reserved | DMAMUXRST | DMA1RST | DMA0RST | Reserved | | | | |
| | | rw | | | | rw | | rw | rw | rw | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Reserved | USBHS0RST | Reserved | ENET1RST |
|---|---|---|---|
| | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | USBHS1RST | USBHS1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USBHS1 |
| 28:26 | Reserved | Must be kept at reset value. |
| 25 | ENET0RST | Ethernet0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the Ethernet0 |
| 24 | Reserved | Must be kept at reset value. |
| 23 | DMAMUXRST | DMAMUX reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DMAMUX |
| 22 | DMA1RST | DMA1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DMA1 |
| 21 | DMA0RST | DMA0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DMA0 |
| 20:15 | Reserved | Must be kept at reset value. |
| 14 | USBHS0RST | USBHS0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USBHS0 |
| 13:1 | Reserved | Must be kept at reset value. |
| 0 | ENET1RST | Ethernet1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the Ethernet1 |

### 6.3.6. AHB2 reset register (RCU_AHB2RST)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TMURST | TRNGRST | Reserved | HAURST | CAURST | SDIO1RST | FACRST | DCIRST |
| | | | | | | | | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | TMURST | TMU reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TMU |
| 6 | TRNGRST | TRNG reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TRNG |
| 5 | Reserved | Must be kept at reset value. |
| 4 | HAURST | HAU reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the HAU |
| 3 | CAURST | CAU reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the CAU |
| 2 | SDIO1RST | SDIO1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SDIO1 |
| 1 | FACRST | FAC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the FAC |

| | | |
|---|---|---|
| 0 | DCIRST | DCI reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DCI |

### 6.3.7. AHB3 reset register (RCU_AHB3RST)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | RTDEC1RST | RTDEC0RST | Reserved | OSPI1RST | OSPI0RST | OSPIMRST | MDMARST | SDIO0RST | IPARST | EXMCRST |
| | | | | | | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | RTDEC1RST | RTDEC1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the RTDEC1 |
| 8 | RTDEC0RST | RTDEC0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the RTDEC0 |
| 7 | Reserved | Must be kept at reset value. |
| 6 | OSPI1RST | OSPI1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the OSPI1 |
| 5 | OSPI0RST | OSPI0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the OSPI0 |
| 4 | OSPIMRST | OSPIM reset<br>This bit is set and reset by software.<br>0: No reset |

| | | |
|---|---|---|
| | | 1: Reset the OSPIM |
| 3 | MDMARST | MDMA reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the MDMA |
| 2 | SDIO0RST | SDIO0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SDIO0 |
| 1 | IPARST | IPA reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the IPA |
| 0 | EXMCRST | EXMC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the EXMC |

### 6.3.8. AHB4 reset register (RCU_AHB4RST)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HWSEMRST | CRCRST | Reserved | | | | PKRST | PJRST | PHRST | PGRST | PFRST | PERST | PDRST | PCRST | PBRST | PARST |
| rw | rw | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | HWSEMRST | HWSEM reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the HWSEM |
| 14 | CRCRST | CRC reset<br>This bit is set and reset by software.<br>0: No reset |

| | | |
|---|---|---|
| | | 1: Reset the CRC |
| 13:10 | Reserved | Must be kept at reset value. |
| 9 | PKRST | GPIO port K reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port K |
| 8 | PJRST | GPIO port J reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port J |
| 7 | PHRST | GPIO port H reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port H |
| 6 | PGRST | GPIO port G reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port G |
| 5 | PFRST | GPIO port F reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port F |
| 4 | PERST | GPIO port E reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port E |
| 3 | PDRST | GPIO port D reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port D |
| 2 | PCRST | GPIO port C reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port C |
| 1 | PBRST | GPIO port B reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port B |
| 0 | PARST | GPIO port A reset |

This bit is set and reset by software.

0: No reset

1: Reset the GPIO port A

### 6.3.9. APB1 reset register (RCU_APB1RST)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UART7RST | UART6RST | DACRST | DACHOLDRST | CTCRST | Reserved | | I2C3RST | I2C2RST | I2C1RST | I2C0RST | UART4RST | UART3RST | USART2RST | USART1RST | MDIORST |
| rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPI2RST | SPI1RST | RSPDIFRST | Reserved | TIMER51RST | TIMER50RST | TIMER31RST | TIMER30RST | TIMER23RST | TIMER22RST | TIMER6RST | TIMER5RST | TIMER4RST | TIMER3RST | TIMER2RST | TIMER1RST |
| rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UART7RST | UART7 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART7 |
| 30 | UART6RST | UART6 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART6 |
| 29 | DACRST | DAC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DAC |
| 28 | DACHOLDRST | DAC hold clock reset<br>This bit is set and reset by software. The hold clock source is IRC32K.<br>0: No reset<br>1: Reset the hold clock |
| 27 | CTCRST | CTC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the CTC |
| 26:25 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 24 | I2C3RST | I2C3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C3 |
| 23 | I2C2RST | I2C2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C2 |
| 22 | I2C1RST | I2C1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C1 |
| 21 | I2C0RST | I2C0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C0 |
| 20 | UART4RST | UART4 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART4 |
| 19 | UART3RST | UART3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART3 |
| 18 | USART2RST | USART2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART2 |
| 17 | USART1RST | USART1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART1 |
| 16 | MDIORST | MDIO reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the MDIO |
| 15 | SPI2RST | SPI2 reset<br>This bit is set and reset by software.<br>0: No reset |

| | | |
|---|---|---|
| | | 1: Reset the SPI2 |
| 14 | SPI1RST | SPI1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI1 |
| 13 | RSPDIFRST | RSPDIF reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the RSPDIF |
| 12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51RST | TIMER51 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER51 |
| 10 | TIMER50RST | TIMER50 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER50 |
| 9 | TIMER31RST | TIMER31 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER31 |
| 8 | TIMER30RST | TIMER30 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER30 |
| 7 | TIMER23RST | TIMER23 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER23 |
| 6 | TIMER22RST | TIMER22 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER22 |
| 5 | TIMER6RST | TIMER6 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER6 |
| 4 | TIMER5RST | TIMER5 reset |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER5 |
| 3 | TIMER4RST | TIMER4 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER4 |
| 2 | TIMER3RST | TIMER3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER3 |
| 1 | TIMER2RST | TIMER2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER2 |
| 0 | TIMER1RST | TIMER1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER1 |

### 6.3.10. APB2 reset register (RCU_APB2RST)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSELRST | EDOUTRST | TIMER44RST | TIMER43RST | TIMER42RST | TIMER41RST | TIMER40RST | SAI2RST | SAI1RST | SAI0RST | SPI5RST | SPI4RST | HPDFRST | TIMER16RST | TIMER15RST | TIMER14RST |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI3RST | SPI0RST | Reserved | ADC2RST | ADC1RST | ADC0RST | Reserved | | USART5RST | USART0RST | Reserved | | TIMER7RST | TIMER0RST |
| | | rw | rw | | rw | rw | rw | | | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TRIGSELRST | TRIGSEL reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TRIGSEL |
| 30 | EDOUTRST | EDOUT reset<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: No reset<br>1: Reset the EDOUT |
| 29 | TIMER44RST | TIMER44 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER44 |
| 28 | TIMER43RST | TIMER43 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER43 |
| 27 | TIMER42RST | TIMER42 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER42 |
| 26 | TIMER41RST | TIMER41 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER41 |
| 25 | TIMER40RST | TIMER40 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER40 |
| 24 | SAI2RST | SAI2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SAI2 |
| 23 | SAI1RST | SAI1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SAI1 |
| 22 | SAI0RST | SAI0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SAI0 |
| 21 | SPI5RST | SPI5 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI5 |
| 20 | SPI4RST | SPI4 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI4 |
| 19 | HPDFRST | HPDF reset |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: No reset<br>1: Reset the HPDF |
| 18 | TIMER16RST | TIMER16 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER16 |
| 17 | TIMER15RST | TIMER15 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER15R |
| 16 | TIMER14RST | TIMER14 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER14 |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SPI3RST | SPI3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI3 |
| 12 | SPI0RST | SPI0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI0 |
| 11 | Reserved | Must be kept at reset value. |
| 10 | ADC2RST | ADC2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the all ADC2 |
| 9 | ADC1RST | ADC1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the all ADC1 |
| 8 | ADC0RST | ADC0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the all ADC0 |
| 7:6 | Reserved | Must be kept at reset value. |

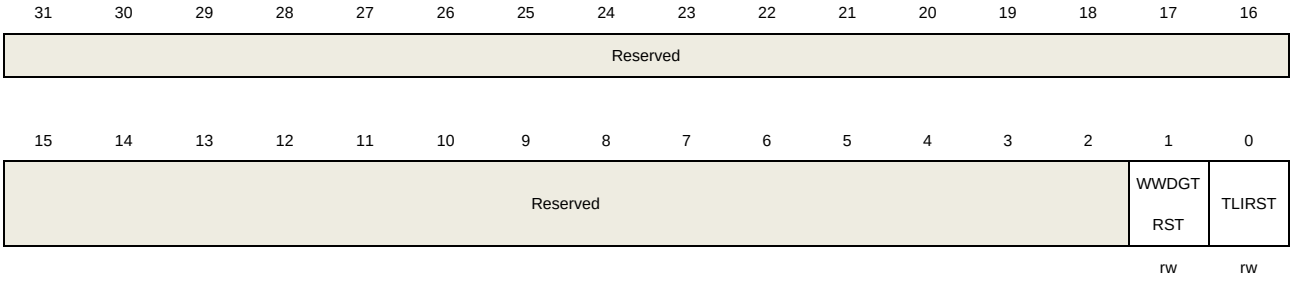
| | | |
|---|---|---|
| 5 | USART5RST | USART5 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART5 |
| 4 | USART0RST | USART0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART0 |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | TIMER7RST | TIMER7 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER7 |
| 0 | TIMER0RST | TIMER0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER0 |

### 6.3.11. APB3 reset register (RCU_APB3RST)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WWDGTRST | WWDGT reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the WWDGT |
| 0 | TLIRST | TLI reset<br>This bit is set and reset by software.<br>0: No reset |

1: Reset the TLI

### 6.3.12. APB4 reset register (RCU_APB4RST)

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | PMURST | LPDTSRST | VREFRST | CMPRST | SYSCFGRST |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | PMURST | PMU reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the PMU |
| 3 | LPDTSRST | LPDTS reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the LPDTS |
| 2 | VREFRST | VREF reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the VREF |
| 1 | CMPRST | CMP reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the CMP |
| 0 | SYSCFGRST | SYSCFG reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SYSCFG |

### 6.3.13. AHB1 enable register (RCU_AHB1EN)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USBHS1ULPIEN | USBHS1EN | ENET0PTPEN | ENET0RXEN | ENET0TXEN | ENET0EN | Reserved | DMAMUXEN | DMA1EN | DMA0EN | Reserved | | | | |
| | rw | rw | rw | rw | rw | rw | | rw | rw | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USBHS0ULPIEN | USBHS0EN | Reserved | | | | | | | | | | ENET1PTPEN | ENET1RXEN | ENET1TXEN | ENET1EN |
| rw | rw | | | | | | | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | USBHS1ULPIEN | USBHS1 ULPI clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS1 ULPI clock<br>1: Enabled USBHS1 ULPI clock |
| 29 | USBHS1EN | USBHS1 clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS1 clock<br>1: Enabled USBHS1 clock |
| 28 | ENET0PTPEN | Ethernet0 PTP clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 PTP clock<br>1: Enabled Ethernet0 PTP clock |
| 27 | ENET0RXEN | Ethernet0 RX clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 RX clock<br>1: Enabled Ethernet0 RX clock |
| 26 | ENET0TXEN | Ethernet0 TX clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 TX clock<br>1: Enabled Ethernet0 TX clock |
| 25 | ENET0EN | Ethernet0 clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 clock<br>1: Enabled Ethernet0 clock |
| 24 | Reserved | Must be kept at reset value. |
| 23 | DMAMUXEN | DMAMUX clock enable<br>This bit is set and reset by software.<br>0: Disabled DMAMUX clock |

| | | |
|---|---|---|
| | | 1: Enabled DMAMUX clock |
| 22 | DMA1EN | DMA1 clock enable<br>This bit is set and reset by software.<br>0: Disabled DMA1 clock<br>1: Enabled DMA1 clock |
| 21 | DMA0EN | DMA0 clock enable<br>This bit is set and reset by software.<br>0: Disabled DMA0 clock<br>1: Enabled DMA0 clock |
| 20:16 | Reserved | Must be kept at reset value. |
| 15 | USBHS0ULPIEN | USBHS0 ULPI clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS0 ULPI clock<br>1: Enabled USBHS0 ULPI clock |
| 14 | USBHS0EN | USBHS0 clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS0 clock<br>1: Enabled USBHS0 clock |
| 13:4 | Reserved | Must be kept at reset value. |
| 3 | ENET1PTPEN | Ethernet1 PTP clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 PTP clock<br>1: Enabled Ethernet1 PTP clock |
| 2 | ENET1RXEN | Ethernet1 RX clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 RX clock<br>1: Enabled Ethernet1 RX clock |
| 1 | ENET1TXEN | Ethernet1 TX clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 TX clock<br>1: Enabled Ethernet1 TX clock |
| 0 | ENET1EN | Ethernet1 clock enable<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 clock<br>1: Enabled Ethernet1 clock |

### 6.3.14. AHB2 enable register (RCU_AHB2EN)

Address offset: 0x34

Reset value: 0x0000 0100

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | RAMECCMU1EN | TMUEN | TRNGEN | Reserved | HAUEN | CAUEN | SDIO1EN | FACEN | DCIEN |
| | | | | | | | rw | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | RAMECCMU1EN | RAMECCMU1 clock enable<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU1 clock<br>1: Enabled RAMECCMU1 clock |
| 7 | TMUEN | TMU clock enable<br>This bit is set and reset by software.<br>0: Disabled TMU clock<br>1: Enabled TMU clock |
| 6 | TRNGEN | TRNG clock enable<br>This bit is set and reset by software.<br>0: Disabled TRNG clock<br>1: Enabled TRNG clock |
| 5 | Reserved | Must be kept at reset value. |
| 4 | HAUEN | HAU clock enable<br>This bit is set and reset by software.<br>0: Disabled HAU clock<br>1: Enabled HAU clock |
| 3 | CAUEN | CAU clock enable<br>This bit is set and reset by software.<br>0: Disabled CAU clock<br>1: Enabled CAU clock |
| 2 | SDIO1EN | SDIO1 clock enable<br>This bit is set and reset by software.<br>0: Disabled SDIO1 clock<br>1: Enabled SDIO1 clock |
| 1 | FACEN | FAC clock enable<br>This bit is set and reset by software.<br>0: Disabled FAC clock |

| | | |
|---|---|---|
| | | 1: Enabled FAC clock |
| 0 | DCIEN | DCI clock enable<br>This bit is set and reset by software.<br>0: Disabled DCI clock<br>1: Enabled DCI clock |

### 6.3.15. AHB3 enable register (RCU_AHB3EN)

Address offset: 0x38

Reset value: 0x0000 8400

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPUEN | Reserved | | | | RAMECCMU0EN | RTDEC1EN | RTDEC0EN | Reserved | OSPI1EN | OSPI0EN | OSPIMEN | MDMAEN | SDIO0EN | IPAEN | EXMCEN |
| rw | | | | | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | CPUEN | CPU clock enable<br>This bit is set and reset by software.<br>0: Disabled CPU clock<br>1: Enabled CPU clock |
| 14:11 | Reserved | Must be kept at reset value. |
| 10 | RAMECCMU0EN | RAMECCMU0 clock enable<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU0 clock<br>1: Enabled RAMECCMU0 clock |
| 9 | RTDEC1EN | RTDEC1 clock enable<br>This bit is set and reset by software.<br>0: Disabled RTDEC1 clock<br>1: Enabled RTDEC1 clock |
| 8 | RTDEC0EN | RTDEC0 clock enable<br>This bit is set and reset by software.<br>0: Disabled RTDEC0 clock<br>1: Enabled RTDEC0 clock |
| 7 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 6 | OSPI1EN | OSPI1 clock enable<br>This bit is set and reset by software.<br>0: Disabled OSPI1 clock<br>1: Enabled OSPI1 clock |
| 5 | OSPI0EN | OSPI0 clock enable<br>This bit is set and reset by software.<br>0: Disabled OSPI0 clock<br>1: Enabled OSPI0 clock |
| 4 | OSPIMEN | OSPIM clock enable<br>This bit is set and reset by software.<br>0: Disabled OSPIM clock<br>1: Enabled OSPIM clock |
| 3 | MDMAEN | MDMA clock enable<br>This bit is set and reset by software.<br>0: Disabled MDMA clock<br>1: Enabled MDMA clock |
| 2 | SDIO0EN | SDIO0 clock enable<br>This bit is set and reset by software.<br>0: Disabled SDIO0 clock<br>1: Enabled SDIO0 clock |
| 1 | IPAEN | IPA clock enable<br>This bit is set and reset by software.<br>0: Disabled IPA clock<br>1: Enabled IPA clock |
| 0 | EXMCEN | EXMC clock enable<br>This bit is set and reset by software.<br>0: Disabled EXMC clock<br>1: Enabled EXMC clock |

### 6.3.16. AHB4 enable register (RCU_AHB4EN)

Address offset: 0x3C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HWSEMEN | CRCEN | BKPSRAMEN | Reserved | | | PKEN | PJEN | PHEN | PGEN | PFEN | PEEN | PDEN | PCEN | PBEN | PAEN |

rw rw rw rw rw rw rw rw rw rw rw rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | HWSEMEN | HWSEM clock enable<br>This bit is set and reset by software.<br>0: Disabled HWSEM clock<br>1: Enabled HWSEM clock |
| 14 | CRCEN | CRC clock enable<br>This bit is set and reset by software.<br>0: Disabled CRC clock<br>1: Enabled CRC clock |
| 13 | BKPSRAMEN | Backup SRAM clock enable<br>This bit is set and reset by software.<br>0: Disabled Backup SRAM clock<br>1: Enabled Backup SRAM clock |
| 12:10 | Reserved | Must be kept at reset value. |
| 9 | PKEN | GPIO port K clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port K clock<br>1: Enabled GPIO port K clock |
| 8 | PJEN | GPIO port J clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port J clock<br>1: Enabled GPIO port J clock |
| 7 | PHEN | GPIO port H clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port H clock<br>1: Enabled GPIO port H clock |
| 6 | PGEN | GPIO port G clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port G clock<br>1: Enabled GPIO port G clock |
| 5 | PFEN | GPIO port F clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port F clock<br>1: Enabled GPIO port F clock |
| 4 | PEEN | GPIO port E clock enable<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled GPIO port E clock<br>1: Enabled GPIO port E clock |
| 3 | PDEN | GPIO port D clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port D clock<br>1: Enabled GPIO port D clock |
| 2 | PCEN | GPIO port C clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port C clock<br>1: Enabled GPIO port C clock |
| 1 | PBEN | GPIO port B clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port B clock<br>1: Enabled GPIO port B clock |
| 0 | PAEN | GPIO port A clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port A clock<br>1: Enabled GPIO port A clock |

### 6.3.17. APB1 enable register (RCU_APB1EN)

Address offset: 0x40

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UART7EN | UART6EN | DACEN | DACHOLDEN | CTCEN | Reserved | | I2C3EN | I2C2EN | I2C1EN | I2C0EN | UART4EN | UART3EN | USART2EN | USART1EN | MDIOEN |
| rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPI2EN | SPI1EN | RSPDIFEN | Reserved | TIMER51EN | TIMER50EN | TIMER31EN | TIMER30EN | TIMER23EN | TIMER22EN | TIMER6EN | TIMER5EN | TIMER4EN | TIMER3EN | TIMER2EN | TIMER1EN |
| rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UART7EN | UART7 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART7 clock<br>1: Enabled UART7 clock |
| 30 | UART6EN | UART6 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART6 clock |

| | | |
|---|---|---|
| | | 1: Enabled UART6 clock |
| 29 | DACEN | DAC clock enable<br>This bit is set and reset by software.<br>0: Disabled DAC clock<br>1: Enabled DAC clock |
| 28 | DACHOLDEN | DAC hold clock enable<br>This bit is set and reset by software. The hold clock source is IRC32K.<br>0: Disabled DAC hold clock<br>1: Enabled DAC hold clock |
| 27 | CTCEN | CTC clock enable<br>This bit is set and reset by software.<br>0: Disabled CTC clock<br>1: Enabled CTC clock |
| 26:25 | Reserved | Must be kept at reset value. |
| 24 | I2C3EN | I2C3 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C3 clock<br>1: Enabled I2C3 clock |
| 23 | I2C2EN | I2C2 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C2 clock<br>1: Enabled I2C2 clock |
| 22 | I2C1EN | I2C1 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C1 clock<br>1: Enabled I2C1 clock |
| 21 | I2C0EN | I2C0 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C0 clock<br>1: Enabled I2C0 clock |
| 20 | UART4EN | UART4 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART4 clock<br>1: Enabled UART4 clock |
| 19 | UART3EN | UART3 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART3 clock<br>1: Enabled UART3 clock |
| 18 | USART2EN | USART2 clock enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled USART2 clock<br>1: Enabled USART2 clock |
| 17 | USART1EN | USART1 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART1 clock<br>1: Enabled USART1 clock |
| 16 | MDIOEN | MDIO clock enable<br>This bit is set and reset by software.<br>0: Disabled MDIO clock<br>1: Enabled MDIO clock |
| 15 | SPI2EN | SPI2 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI2 clock<br>1: Enabled SPI2 clock |
| 14 | SPI1EN | SPI1 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI1 clock<br>1: Enabled SPI1 clock |
| 13 | RSPDIFEN | RSPDIF clock enable<br>This bit is set and reset by software.<br>0: Disabled RSPDIF clock<br>1: Enabled RSPDIF clock |
| 12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51EN | TIMER51 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER51 clock<br>1: Enabled TIMER51 clock |
| 10 | TIMER50EN | TIMER50 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER50 clock<br>1: Enabled TIMER50 clock |
| 9 | TIMER31EN | TIMER31 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER31 clock<br>1: Enabled TIMER31 clock |
| 8 | TIMER30EN | TIMER30 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER30 clock |

| | | |
|---|---|---|
| | | 1: Enabled TIMER30 clock |
| 7 | TIMER23EN | TIMER23 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER23 clock<br>1: Enabled TIMER23 clock |
| 6 | TIMER22EN | TIMER22 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER22 clock<br>1: Enabled TIMER22 clock |
| 5 | TIMER6EN | TIMER6 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER6 clock<br>1: Enabled TIMER6 clock |
| 4 | TIMER5EN | TIMER5 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER5 clock<br>1: Enabled TIMER5 clock |
| 3 | TIMER4EN | TIMER4 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER4 clock<br>1: Enabled TIMER4 clock |
| 2 | TIMER3EN | TIMER3 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER3 clock<br>1: Enabled TIMER3 clock |
| 1 | TIMER2EN | TIMER2 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER2 clock<br>1: Enabled TIMER2 clock |
| 0 | TIMER1EN | TIMER1 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER1 clock<br>1: Enabled TIMER1 clock |

### 6.3.18. APB2 enable register (RCU_APB2EN)

Address offset: 0x44

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSELEN | EDOUTEN | TIMER44EN | TIMER43EN | TIMER42EN | TIMER41EN | TIMER40EN | SAI2EN | SAI1EN | SAI0EN | SPI5EN | SPI4EN | HPDFEN | TIMER16EN | TIMER15EN | TIMER14EN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI3EN | SPI0EN | Reserved | ADC2EN | ADC1EN | ADC0EN | Reserved | | USART5EN | USART0EN | Reserved | | TIMER7EN | TIMER0EN |
| | | rw | rw | | rw | rw | rw | | | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TRIGSELEN | TRIGSEL clock enable<br>This bit is set and reset by software.<br>0: Disabled TRIGSEL clock<br>1: Enabled TRIGSEL clock |
| 30 | EDOUTEN | EDOUT clock enable<br>This bit is set and reset by software.<br>0: Disabled EDOUT clock<br>1: Enabled EDOUT clock |
| 29 | TIMER44EN | TIMER44 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER44 clock<br>1: Enabled TIMER44 clock |
| 28 | TIMER43EN | TIMER43 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER43 clock<br>1: Enabled TIMER43 clock |
| 27 | TIMER42EN | TIMER42 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER42 clock<br>1: Enabled TIMER42 clock |
| 26 | TIMER41EN | TIMER41 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER41 clock<br>1: Enabled TIMER41 clock |
| 25 | TIMER40EN | TIMER40 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER40 clock<br>1: Enabled TIMER40 clock |
| 24 | SAI2EN | SAI2 clock enable<br>This bit is set and reset by software.<br>0: Disabled SAI2 clock |

| | | |
|---|---|---|
| | | 1: Enabled SAI2 clock |
| 23 | SAI1EN | SAI1 clock enable<br>This bit is set and reset by software.<br>0: Disabled SAI1 clock<br>1: Enabled SAI1 clock |
| 22 | SAI0EN | SAI0 clock enable<br>This bit is set and reset by software.<br>0: Disabled SAI0 clock<br>1: Enabled SAI0 clock |
| 21 | SPI5EN | SPI5 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI5 clock<br>1: Enabled SPI5 clock |
| 20 | SPI4EN | SPI4 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI4 clock<br>1: Enabled SPI4 clock |
| 19 | HPDFEN | HPDF clock enable<br>This bit is set and reset by software.<br>0: Disabled HPDF clock<br>1: Enabled HPDF clock |
| 18 | TIMER16EN | TIMER16 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER16 clock<br>1: Enabled TIMER16 clock |
| 17 | TIMER15EN | TIMER15 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER15 clock<br>1: Enabled TIMER15 clock |
| 16 | TIMER14EN | TIMER14 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER14 clock<br>1: Enabled TIMER14 clock |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SPI3EN | SPI3 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI3 clock<br>1: Enabled SPI3 clock |
| 12 | SPI0EN | SPI0 clock enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled SPI0 clock<br>1: Enabled SPI0 clock |
| 11 | Reserved | Must be kept at reset value. |
| 10 | ADC2EN | ADC2 clock enable<br>This bit is set and reset by software.<br>0: Disabled ADC2 clock<br>1: Enabled ADC2 clock |
| 9 | ADC1EN | ADC1 clock enable<br>This bit is set and reset by software.<br>0: Disabled ADC1 clock<br>1: Enabled ADC1 clock |
| 8 | ADC0EN | ADC0 clock enable<br>This bit is set and reset by software.<br>0: Disabled ADC0 clock<br>1: Enabled ADC0 clock |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | USART5EN | USART5 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART5 clock<br>1: Enabled USART5 clock |
| 4 | USART0EN | USART0 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART0 clock<br>1: Enabled USART0 clock |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | TIMER7EN | TIMER7 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER7 clock<br>1: Enabled TIMER7 clock |
| 0 | TIMER0EN | TIMER0 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER0 clock<br>1: Enabled TIMER0 clock |

### 6.3.19. APB3 enable register (RCU_APB3EN)

Address offset: 0x48

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | WWDGTEN | TLIEN |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WWDGTEN | WWDGT clock enable<br>This bit is set and reset by software.<br>0: Disabled WWDGT clock<br>1: Enabled WWDGT clock |
| 0 | TLIEN | TLI clock enable<br>This bit is set and reset by software.<br>0: Disabled TLI clock<br>1: Enabled TLI clock |

## 6.3.20. APB4 enable register (RCU_APB4EN)

Address offset: 0x4C

Reset value: 0x0000 0010

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | PMUEN | LPDTSEN | VREFEN | CMPEN | SYSCFGEN |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | PMUEN | PMU clock enable<br>This bit is set and reset by software.<br>0: Disabled PMU clock<br>1: Enabled PMU clock |
| 3 | LPDTSEN | LPDTS clock enable<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled LPDTS clock<br>1: Enabled LPDTS clock |
| 2 | VREFEN | VREF clock enable<br>This bit is set and reset by software.<br>0: Disabled VREF clock<br>1: Enabled VREF clock |
| 1 | CMPEN | CMP clock enable<br>This bit is set and reset by software.<br>0: Disabled CMP clock<br>1: Enabled CMP clock |
| 0 | SYSCFGEN | SYSCFG clock enable<br>This bit is set and reset by software.<br>0: Disabled SYSCFG clock<br>1: Enabled SYSCFG clock |

## 6.3.21. AHB1 sleep mode enable register (RCU_AHB1SPEN)

Address offset: 0x50

Reset value: 0x7EE3 C00F

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USBHS1ULPISPEN | USBHS1SPEN | ENET0PTPSPEN | ENET0RXSPEN | ENET0TXSPEN | ENET0SPEN | Reserved | DMAMUXSPEN | DMA1SPEN | DMA0SPEN | Reserved | | | SRAM1SPEN | SRAM0SPEN |
| | rw | rw | rw | rw | rw | rw | | rw | rw | rw | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USBHS0ULPISPEN | USBHS0SPEN | Reserved | | | | | | | | | | ENET1PTPSPEN | ENET1RXSPEN | ENET1TXSPEN | ENET1SPEN |
| rw | rw | | | | | | | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | USBHS1ULPISPEN | USBHS1 ULPI clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS1 ULPI clock when sleep mode<br>1: Enabled USBHS1 ULPI clock when sleep mode |
| 29 | USBHS1SPEN | USBHS1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS1 clock when sleep mode<br>1: Enabled USBHS1 clock when sleep mode |
| 28 | ENET0PTPSPEN | Ethernet0 PTP clock enable when sleep mode |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled Ethernet0 PTP clock when sleep mode<br>1: Enabled Ethernet0 PTP clock when sleep mode |
| 27 | ENET0RXSPEN | Ethernet0 RX clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 RX clock when sleep mode<br>1: Enabled Ethernet0 RX clock when sleep mode |
| 26 | ENET0TXSPEN | Ethernet0 TX clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 TX clock when sleep mode<br>1: Enabled Ethernet0 TX clock when sleep mode |
| 25 | ENET0SPEN | Ethernet0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet0 clock when sleep mode<br>1: Enabled Ethernet0 clock when sleep mode |
| 24 | Reserved | Must be kept at reset value. |
| 23 | DMAMUXSPEN | DMAMUX clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DMAMUX clock when sleep mode<br>1: Enabled DMAMUX clock when sleep mode |
| 22 | DMA1SPEN | DMA1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DMA1 clock when sleep mode<br>1: Enabled DMA1 clock when sleep mode |
| 21 | DMA0SPEN | DMA0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DMA0 clock when sleep mode<br>1: Enabled DMA0 clock when sleep mode |
| 20:18 | Reserved | Must be kept at reset value. |
| 17 | SRAM1SPEN | SRAM1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SRAM1 clock when sleep mode<br>1: Enabled SRAM1 clock when sleep mode |
| 16 | SRAM0SPEN | SRAM0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SRAM0 clock when sleep mode<br>1: Enabled SRAM0 clock when sleep mode |
| 15 | USBHS0ULPISPEN | USBHS0 ULPI clock enable when sleep mode<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled USBHS0 ULPI clock when sleep mode<br>1: Enabled USBHS0 ULPI clock when sleep mode |
| 14 | USBHS0SPEN | USBHS0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS0 clock when sleep mode<br>1: Enabled USBHS0 clock when sleep mode |
| 13:4 | Reserved | Must be kept at reset value. |
| 3 | ENET1PTPSPEN | Ethernet1 PTP clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 PTP clock when sleep mode<br>1: Enabled Ethernet1 PTP clock when sleep mode |
| 2 | ENET1RXSPEN | Ethernet1 RX clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 RX clock when sleep mode<br>1: Enabled Ethernet1 RX clock when sleep mode |
| 1 | ENET1TXSPEN | Ethernet1 TX clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 TX clock when sleep mode<br>1: Enabled Ethernet1 TX clock when sleep mode |
| 0 | ENET1SPEN | Ethernet1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Ethernet1 clock when sleep mode<br>1: Enabled Ethernet1 clock when sleep mode |

## 6.3.22. AHB2 sleep mode enable register (RCU_AHB2SPEN)

Address offset: 0x54

Reset value: 0x0000 01DF

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | RAMECCMU1SPEN | TMUSPEN | TRNGSPEN | Reserved | HAUSPEN | CAUSPEN | SDIO1SPEN | FACSPEN | DCISPEN |
| | | | | | | | rw | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | RAMECCMU1SPEN | RAMECCMU1 clock enable when sleep mode |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled RAMECCMU1 clock when sleep mode<br>1: Enabled RAMECCMU1 clock when sleep mode |
| 7 | TMUSPEN | TMU clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TMU clock when sleep mode<br>1: Enabled TMU clock when sleep mode |
| 6 | TRNGSPEN | TRNG clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TRNG clock when sleep mode<br>1: Enabled TRNG clock when sleep mode |
| 5 | Reserved | Must be kept at reset value. |
| 4 | HAUSPEN | HAU enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled HAU clock when sleep mode<br>1: Enabled HAU clock when sleep mode |
| 3 | CAUSPEN | CAU clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CAU clock when sleep mode<br>1: Enabled CAU clock when sleep mode |
| 2 | SDIO1SPEN | SDIO1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SDIO1 clock when sleep mode<br>1: Enabled SDIO1 clock when sleep mode |
| 1 | FACSPEN | FAC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled FAC clock when sleep mode<br>1: Enabled FAC clock when sleep mode |
| 0 | DCISPEN | DCI clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DCI clock when sleep mode<br>1: Enabled DCI clock when sleep mode |

### 6.3.23. AHB3 sleep mode enable register (RCU_AHB3SPEN)

Address offset: 0x58

Reset value: 0x0000 C77F

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMCSPEN | AXISRAMSPEN | Reserved | | | RAMECCMU0SPEN | RTDEC1SPEN | RTDEC0SPEN | Reserved | OSPI1SPEN | OSPI0SPEN | OSPIMSPEN | MDMASPEN | SDIO0SPEN | IPASPEN | EXMCSPEN |
| rw | rw | | | | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | FMCSPEN | FMC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled FMC clock when sleep mode<br>1: Enabled FMC clock when sleep mode |
| 14 | AXISRAMSPEN | AXI SRAM clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled AXI SRAM clock when sleep mode<br>1: Enabled AXI SRAM clock when sleep mode |
| 13:11 | Reserved | Must be kept at reset value. |
| 10 | RAMECCMU0SPEN | RAMECCMU0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU0 clock when sleep mode<br>1: Enabled RAMECCMU0 clock when sleep mode |
| 9 | RTDEC1SPEN | RTDEC1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled RTDEC1 clock when sleep mode<br>1: Enabled RTDEC1 clock when sleep mode |
| 8 | RTDEC0SPEN | RTDEC0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled RTDEC0 clock when sleep mode<br>1: Enabled RTDEC0 clock when sleep mode |
| 7 | Reserved | Must be kept at reset value. |
| 6 | OSPI1SPEN | OSPI1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled OSPI1 clock when sleep mode<br>1: Enabled OSPI1 clock when sleep mode |
| 5 | OSPI0SPEN | OSPI0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled OSPI0 clock when sleep mode<br>1: Enabled OSPI0 clock when sleep mode |

| | | |
|---|---|---|
| 4 | OSPIMSPEN | OSPIM clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled OSPIM clock when sleep mode<br>1: Enabled OSPIM clock when sleep mode |
| 3 | MDMASPEN | MDMA clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled MDMA clock when sleep mode<br>1: Enabled MDMA clock when sleep mode |
| 2 | SDIO0SPEN | SDIO0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SDIO0 clock when sleep mode<br>1: Enabled SDIO0 clock when sleep mode |
| 1 | IPASPEN | IPA clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled IPA clock when sleep mode<br>1: Enabled IPA clock when sleep mode |
| 0 | EXMCSPEN | EXMC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled EXMC clock when sleep mode<br>1: Enabled EXMC clock when sleep mode |

### 6.3.24. AHB4 sleep mode enable register (RCU_AHB4SPEN)

Address offset: 0x5C

Reset value: 0x0000 63FF

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | CRCSPEN | BKPSRAMSPEN | Reserved | | | PKSPEN | PJSPEN | PHSPEN | PGSPEN | PFSPEN | PESPEN | PDSPEN | PCSPEN | PBSPEN | PASPEN |
| | rw | rw | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | CRCSPEN | CRC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CRC clock when sleep mode<br>1: Enabled CRC clock when sleep mode |

| | | |
|---|---|---|
| 13 | BKPSRAMSPEN | Backup SRAM clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Backup SRAM clock when sleep mode<br>1: Enabled Backup SRAM clock when sleep mode |
| 12:10 | Reserved | Must be kept at reset value. |
| 9 | PKSPEN | GPIO port K clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port K clock when sleep mode<br>1: Enabled GPIO port K clock when sleep mode |
| 8 | PJSPEN | GPIO port J clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port J clock when sleep mode<br>1: Enabled GPIO port J clock when sleep mode |
| 7 | PHSPEN | GPIO port H clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port H clock when sleep mode<br>1: Enabled GPIO port H clock when sleep mode |
| 6 | PGSPEN | GPIO port G clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port G clock when sleep mode<br>1: Enabled GPIO port G clock when sleep mode |
| 5 | PFSPEN | GPIO port F clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port F clock when sleep mode<br>1: Enabled GPIO port F clock when sleep mode |
| 4 | PESPEN | GPIO port E clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port E clock when sleep mode<br>1: Enabled GPIO port E clock when sleep mode |
| 3 | PDSPEN | GPIO port D clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port D clock when sleep mode<br>1: Enabled GPIO port D clock when sleep mode |
| 2 | PCSPEN | GPIO port C clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port C clock when sleep mode<br>1: Enabled GPIO port C clock when sleep mode |
| 1 | PBSPEN | GPIO port B clock enable when sleep mode<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled GPIO port B clock when sleep mode<br>1: Enabled GPIO port B clock when sleep mode |
| 0 | PASPEN | GPIO port A clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port A clock when sleep mode<br>1: Enabled GPIO port A clock when sleep mode |

## 6.3.25. APB1 sleep mode enable register (RCU_APB1SPEN)

Address offset: 0x60

Reset value: 0xF9FF EFFF

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UART7SPEN | UART6SPEN | DACSPEN | DACHOLDSPEN | CTCSPEN | Reserved | | I2C3SPEN | I2C2SPEN | I2C1SPEN | I2C0SPEN | UART4SPEN | UART3SPEN | USART2SPEN | USART1SPEN | MDIOSPEN |
| rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPI2SPEN | SPI1SPEN | RSPDIFSPEN | Reserved | TIMER51SPEN | TIMER50SPEN | TIMER31SPEN | TIMER30SPEN | TIMER23SPEN | TIMER22SPEN | TIMER6SPEN | TIMER5SPEN | TIMER4SPEN | TIMER3SPEN | TIMER2SPEN | TIMER1SPEN |
| rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UART7SPEN | UART7 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART7 clock when sleep mode<br>1: Enabled UART7 clock when sleep mode |
| 30 | UART6SPEN | UART6 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART6 clock when sleep mode<br>1: Enabled UART6 clock when sleep mode |
| 29 | DACSPEN | DAC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DAC clock when sleep mode<br>1: Enabled DAC clock when sleep mode |
| 28 | DACHOLDSPEN | DAC hold clock enable when sleep mode<br>This bit is set and reset by software. The hold clock source is IRC32K.<br>0: Disabled DAC hold clock when sleep mode<br>1: Enabled DAC hold clock when sleep mode |
| 27 | CTCSPEN | CTC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CTC clock when sleep mode |

| | | |
|---|---|---|
| | | 1: Enabled CTC clock when sleep mode |
| 26:25 | Reserved | Must be kept at reset value. |
| 24 | I2C3SPEN | I2C3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C3 clock when sleep mode<br>1: Enabled I2C3 clock when sleep mode |
| 23 | I2C2SPEN | I2C2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C2 clock when sleep mode<br>1: Enabled I2C2 clock when sleep mode |
| 22 | I2C1SPEN | I2C1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C1 clock when sleep mode<br>1: Enabled I2C1 clock when sleep mode |
| 21 | I2C0SPEN | I2C0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C0 clock when sleep mode<br>1: Enabled I2C0 clock when sleep mode |
| 20 | UART4SPEN | UART4 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART4 clock when sleep mode<br>1: Enabled UART4 clock when sleep mode |
| 19 | UART3SPEN | UART3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART3 clock when sleep mode<br>1: Enabled UART3 clock when sleep mode |
| 18 | USART2SPEN | USART2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART2 clock when sleep mode<br>1: Enabled USART2 clock when sleep mode |
| 17 | USART1SPEN | USART1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART1 clock when sleep mode<br>1: Enabled USART1 clock when sleep mode |
| 16 | MDIOSPEN | MDIO clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled MDIO clock when sleep mode<br>1: Enabled MDIO clock when sleep mode |
| 15 | SPI2SPEN | SPI2 clock enable when sleep mode |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled SPI2 clock when sleep mode<br>1: Enabled SPI2 clock when sleep mode |
| 14 | SPI1SPEN | SPI1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI1 clock when sleep mode<br>1: Enabled SPI1 clock when sleep mode |
| 13 | RSPDIFSPEN | RSPDIF clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled RSPDIF clock when sleep mode<br>1: Enabled RSPDIF clock when sleep mode |
| 12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51SPEN | TIMER51 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER51 clock when sleep mode<br>1: Enabled TIMER51 clock when sleep mode |
| 10 | TIMER50SPEN | TIMER50 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER50 clock when sleep mode<br>1: Enabled TIMER50 clock when sleep mode |
| 9 | TIMER31SPEN | TIMER31 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER31 clock when sleep mode<br>1: Enabled TIMER31 clock when sleep mode |
| 8 | TIMER30SPEN | TIMER30 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER30 clock when sleep mode<br>1: Enabled TIMER30 clock when sleep mode |
| 7 | TIMER23SPEN | TIMER23 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER23 clock when sleep mode<br>1: Enabled TIMER23 clock when sleep mode |
| 6 | TIMER22SPEN | TIMER22 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER22 clock when sleep mode<br>1: Enabled TIMER22 clock when sleep mode |
| 5 | TIMER6SPEN | TIMER6 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER6 clock when sleep mode |

| | | |
|---|---|---|
| | | 1: Enabled TIMER6 clock when sleep mode |
| 4 | TIMER5SPEN | TIMER5 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER5 clock when sleep mode<br>1: Enabled TIMER5 clock when sleep mode |
| 3 | TIMER4SPEN | TIMER4 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER4 clock when sleep mode<br>1: Enabled TIMER4 clock when sleep mode |
| 2 | TIMER3SPEN | TIMER3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER3 clock when sleep mode<br>1: Enabled TIMER3 clock when sleep mode |
| 1 | TIMER2SPEN | TIMER2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER2 clock when sleep mode<br>1: Enabled TIMER2 clock when sleep mode |
| 0 | TIMER1SPEN | TIMER1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER1 clock when sleep mode<br>1: Enabled TIMER1 clock when sleep mode |

### 6.3.26. APB2 sleep mode enable register (RCU_APB2SPEN)

Address offset: 0x64

Reset value: 0xFFFF 3733

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSELSPEN | EDOUTSPEN | TIMER44SPEN | TIMER43SPEN | TIMER42SPEN | TIMER41SPEN | TIMER40SPEN | SAI2SPEN | SAI1SPEN | SAI0SPEN | SPI5SPEN | SPI4SPEN | HPDFSPEN | TIMER16SPEN | TIMER15SPEN | TIMER14SPEN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI3SPEN | SPI0SPEN | Reserved | ADC2SPEN | ADC1SPEN | ADC0SPEN | Reserved | | USART5SPEN | USART0SPEN | Reserved | | TIMER7SPEN | TIMER0SPEN |
| | | rw | rw | | rw | rw | rw | | | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TRIGSELSPEN | TRIGSEL clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TRIGSEL clock when sleep mode |

| | | |
|---|---|---|
| | | 1: Enabled TRIGSEL clock when sleep mode |
| 30 | EDOUTSPEN | EDOUT clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled EDOUT clock when sleep mode<br>1: Enabled EDOUT clock when sleep mode |
| 29 | TIMER44SPEN | TIMER44 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER44 clock when sleep mode<br>1: Enabled TIMER44 clock when sleep mode |
| 28 | TIMER43SPEN | TIMER43 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER43 clock when sleep mode<br>1: Enabled TIMER43 clock when sleep mode |
| 27 | TIMER42SPEN | TIMER42 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER42 clock when sleep mode<br>1: Enabled TIMER42 clock when sleep mode |
| 26 | TIMER41SPEN | TIMER41 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER41 clock when sleep mode<br>1: Enabled TIMER41 clock when sleep mode |
| 25 | TIMER40SPEN | TIMER40 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER40 clock when sleep mode<br>1: Enabled TIMER40 clock when sleep mode |
| 24 | SAI2SPEN | SAI2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SAI2 clock when sleep mode<br>1: Enabled SAI2 clock when sleep mode |
| 23 | SAI1SPEN | SAI1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SAI1 clock when sleep mode<br>1: Enabled SAI1 clock when sleep mode |
| 22 | SAI0SPEN | SAI0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SAI0 clock when sleep mode<br>1: Enabled SAI0 clock when sleep mode |
| 21 | SPI5SPEN | SPI5 clock enable when sleep mode<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled SPI5 clock when sleep mode<br>1: Enabled SPI5 clock when sleep mode |
| 20 | SPI4SPEN | SPI4 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI4 clock when sleep mode<br>1: Enabled SPI4 clock when sleep mode |
| 19 | HPDFSPEN | HPDF clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled HPDF clock when sleep mode<br>1: Enabled HPDF clock when sleep mode |
| 18 | TIMER16SPEN | TIMER16 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER16 clock when sleep mode<br>1: Enabled TIMER16 clock when sleep mode |
| 17 | TIMER15SPEN | TIMER15 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER15 clock when sleep mode<br>1: Enabled TIMER15 clock when sleep mode |
| 16 | TIMER14SPEN | TIMER14 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER14 clock when sleep mode<br>1: Enabled TIMER14 clock when sleep mode |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SPI3SPEN | SPI3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI3 clock when sleep mode<br>1: Enabled SPI3 clock when sleep mode |
| 12 | SPI0SPEN | SPI0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI0 clock when sleep mode<br>1: Enabled SPI0 clock when sleep mode |
| 11 | Reserved | Must be kept at reset value. |
| 10 | ADC2SPEN | ADC2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled ADC2 clock when sleep mode<br>1: Enabled ADC2 clock when sleep mode |
| 9 | ADC1SPEN | ADC1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled ADC1 clock when sleep mode |

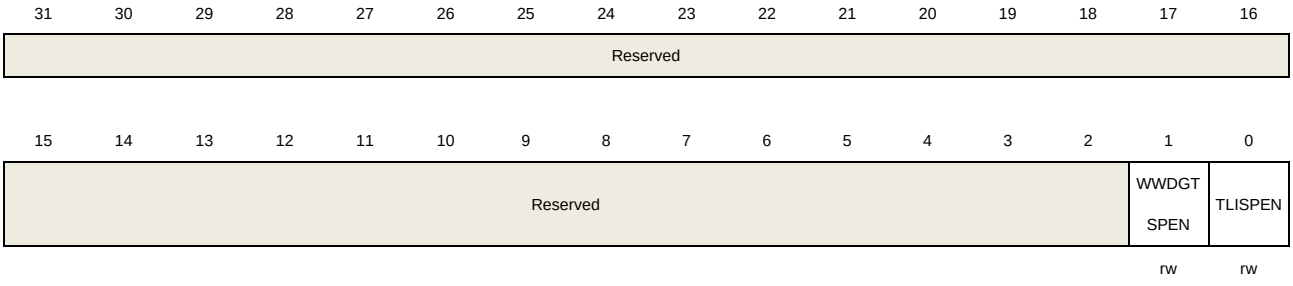
| | | |
|---|---|---|
| | | 1: Enabled ADC1 clock when sleep mode |
| 8 | ADC0SPEN | ADC0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled ADC0 clock when sleep mode<br>1: Enabled ADC0 clock when sleep mode |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | USART5SPEN | USART5 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART5 clock when sleep mode<br>1: Enabled USART5 clock when sleep mode |
| 4 | USART0SPEN | USART0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART0 clock when sleep mode<br>1: Enabled USART0 clock when sleep mode |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | TIMER7SPEN | TIMER7 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER7 clock when sleep mode<br>1: Enabled TIMER7 clock when sleep mode |
| 0 | TIMER0SPEN | TIMER0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER0 clock when sleep mode<br>1: Enabled TIMER0 clock when sleep mode |

### 6.3.27. APB3 sleep mode enable register (RCU_APB3SPEN)

Address offset: 0x68

Reset value: 0x0000 0003

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 1 | WWDGTSPEN | WWDGT clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled WWDGT clock when sleep mode<br>1: Enabled WWDGT clock when sleep mode |
| 0 | TLISPEN | TLI clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TLI clock when sleep mode<br>1: Enabled TLI clock when sleep mode |

### 6.3.28. APB4 sleep mode enable register (RCU_APB4SPEN)

Address offset: 0x6C

Reset value: 0x0000 001F

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | PMUSPEN | LPDTSSPEN | VREFSPEN | CMPSPEN | SYSCFGSPEN |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | PMUSPEN | PMU clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled PMU clock when sleep mode<br>1: Enabled PMU clock when sleep mode |
| 3 | LPDTSSPEN | LPDTS clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled LPDTS clock when sleep mode<br>1: Enabled LPDTS clock when sleep mode |
| 2 | VREFSPEN | VREF clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled VREF clock when sleep mode<br>1: Enabled VREF clock when sleep mode |
| 1 | CMPSPEN | CMPclock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CMPclock when sleep mode<br>1: Enabled CMP clock when sleep mode |

| | | |
|---|---|---|
| 0 | SYSCFGSPEN | SYSCFG clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SYSCFG clock when sleep mode<br>1: Enabled SYSCFG clock when sleep mode |

## 6.3.29. Backup domain control register (RCU_BDCTL)

Address offset: 0x70

Reset value: 0x0000 0018, reset by Backup domain Reset.

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

**Note:** The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the backup domain control register (RCU_BDCTL) are only reset after a backup domain reset. These bits can be modified only when the BKPWEN bit in the power control register (PMU_CTL) is set.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | BKPRST |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTCEN | Reserved | | | | | RTCSRC[1:0] | | Reserved | LCKMD | LCKMEN | LXTALDRI[1:0] | | LXTALBPS | LXTALSTB | LXTALEN |
| rw | | | | | | rw | | | r | rw | rw | | rw | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | BKPRST | Backup domain reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Resets backup domain |
| 15 | RTCEN | RTC clock enable<br>This bit is set and reset by software.<br>0: Disabled RTC clock<br>1: Enabled RTC clock |
| 14:10 | Reserved | Must be kept at reset value. |
| 9:8 | RTCSRC[1:0] | RTC clock entry selection<br>Set and reset by software to control the RTC clock source. Once the RTC clock source has been selected, it cannot be changed anymore unless the backup domain is reset.<br>00: No clock selected<br>01: CK_LXTAL selected as RTC source clock<br>10: CK_IRC32K selected as RTC source clock<br>11: (CK_HXTAL / RTCDIV) selected as RTC source clock, please refer to RTCDIV |

| | | |
|---|---|---|
| | | bits in RCU_CFG0 register. |
| 7 | Reserved | Must be kept at reset value. |
| 6 | LCKMD | LXTAL clock failure detection<br>Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator (LXTAL). It can be clean by disable LCKMEN or disable LXTALEN.<br>0: No failure detected on LXTAL (32 kHz oscillator)<br>1: Failure detected on LXTAL (32 kHz oscillator) |
| 5 | LCKMEN | LXTAL clock monitor enable<br>0: Disable the LXTAL clock monitor<br>1: Enable the LXTAL clock monitor<br>Set by software to enable the clock security system on LXTAL (32 kHz oscillator). LCKMEN should be enabled only on the LXTAL is enabled (LXTALEN bit enabled) and ready (LXTALSTB flag set by hardware).<br>**Note:** Once LCKMEN bit is set, this bit can be reset by backup domain reset or reseting this bit after detecting LXTAL clock failure (LCKMD = 1). |
| 4:3 | LXTALDRI[1:0] | LXTAL drive capability<br>Set and reset by software. Backup domain reset resets this value.<br>00: Lower driving capability<br>01: Medium low driving capability<br>10: Medium high driving capability<br>11: Higher driving capability<br>**Note:** The LXTALDRI is not in bypass mode. |
| 2 | LXTALBPS | LXTAL bypass mode enable<br>Set and reset by software.<br>0: Disable the LXTAL Bypass mode<br>1: Enable the LXTAL Bypass mode |
| 1 | LXTALSTB | Low speed crystal oscillator stabilization flag<br>Set by hardware to indicate if the LXTAL output clock is stable and ready for use.<br>0: LXTAL is not stable<br>1: LXTAL is stable |
| 0 | LXTALEN | LXTAL enable<br>Set and reset by software.<br>0: Disable LXTAL<br>1: Enable LXTAL |

### 6.3.30. Reset source/clock register (RCU_RSTSCK)

Address offset: 0x74

Reset value: 0x0E00 0000, all reset flags reset by power reset only, RSTFC / IRC32KEN reset by system reset.

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPRSTF | WWDGTRSTF | FWDGTRSTF | SWRSTF | PORRSTF | EPRSTF | BORRSTF | RSTFC | Reserved | | | | | | | |
| r | r | r | r | r | r | r | rw | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | IRC32KSTB | IRC32KEN |
| | | | | | | | | | | | | | | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LPRSTF | Low-power reset flag<br>Set by hardware when Deep-sleep /standby reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No Low-power management reset generated<br>1: Low-power management reset generated |
| 30 | WWDGTRSTF | Window watchdog timer reset flag<br>Set by hardware when a window watchdog timer reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No window watchdog reset generated<br>1: Window watchdog reset generated |
| 29 | FWDGTRSTF | Free watchdog timer reset flag<br>Set by hardware when a free watchdog timer reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No free watchdog timer reset generated<br>1: free Watchdog timer reset generated |
| 28 | SWRSTF | Software reset flag<br>Set by hardware when a software reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No software reset generated<br>1: Software reset generated |
| 27 | PORRSTF | Power reset flag<br>Set by hardware when a power reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No Power reset generated<br>1: Power reset generated |
| 26 | EPRSTF | External PIN reset flag<br>Set by hardware when an external pin reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No external pin reset generated<br>1: External pin reset generated |

| | | |
|---|---|---|
| 25 | BORRSTF | BOR reset flag<br>Set by hardware when a BOR reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No BOR reset generated<br>1: BOR reset generated |
| 24 | RSTFC | Reset flag clear<br>This bit is set by software to clear all reset flags.<br>0: Not clear reset flags<br>1: Clear reset flags |
| 23:2 | Reserved | Must be kept at reset value. |
| 1 | IRC32KSTB | IRC32K stabilization flag<br>Set by hardware to indicate if the IRC32K output clock is stable and ready for use.<br>0: IRC32K is not stable<br>1: IRC32K is stable |
| 0 | IRC32KEN | IRC32K enable<br>Set and reset by software.<br>0: Disable IRC32K<br>1: Enable IRC32K |

## 6.3.31. PLL clock additional control register (RCU_PLLADDCTL)

Address offset: 0x80

Reset value: 0xFF81 0101

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLL2PEN | PLL2REN | PLL2QEN | PLL1PEN | PLL1REN | PLL1QEN | PLL0PEN | PLL0REN | PLL0QEN | PLL2Q[6:0] | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | PLL1Q[6:0] | | | | | | | Reserved | PLL0Q[6:0] | | | | | | |
| | rw | | | | | | | | rw | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLL2PEN | PLL2P divider output enable<br>This bit is set and reset by software. The PLL2PEN bit can be written only if the PLL2EN is 0.<br>0: Disable the CK_PLL2P output<br>1: Enable the CK_PLL2P output |
| 30 | PLL2REN | PLL2R divider output enable<br>This bit is set and reset by software. The PLL2REN bit can be written only if the PLL2EN is 0.<br>0: Disable the CK_PLL2R output |

| | | |
|---|---|---|
| | | 1: Enable the CK_PLL2R output |
| 29 | PLL2QEN | PLL2Q divider output enable<br>This bit is set and reset by software. The PLL2QEN bit can be written only if the PLL2EN is 0.<br>0: Disable the CK_PLL2Q output<br>1: Enable the CK_PLL2Q output |
| 28 | PLL1PEN | PLL1P divider output enable<br>This bit is set and reset by software. The PLL1PEN bit can be written only if the PLL1EN is 0.<br>0: Disable the CK_PLL1P output<br>1: Enable the CK_PLL1P output |
| 27 | PLL1REN | PLL1R divider output enable<br>This bit is set and reset by software. The PLL1REN bit can be written only if the PLL1EN is 0.<br>0: Disable the CK_PLL1R output<br>1: Enable the CK_PLL1R output |
| 26 | PLL1QEN | PLL1Q divider output enable<br>This bit is set and reset by software. The PLL1QEN bit can be written only if the PLL1EN is 0.<br>0: Disable the CK_PLL1Q output<br>1: Enable the CK_PLL1Q output |
| 25 | PLL0PEN | PLL0P divider output enable<br>This bit is set and reset by software. The PLL0PEN bit can be written only if the PLL0EN is 0.<br>0: Disable the CK_PLL0P output<br>1: Enable the CK_PLL0P output |
| 24 | PLL0REN | PLL0R divider output enable<br>This bit is set and reset by software. The PLL0REN bit can be written only if the PLL0EN is 0.<br>0: Disable the CK_PLL0R output<br>1: Enable the CK_PLL0R output |
| 23 | PLL0QEN | PLL0Q divider output enable<br>This bit is set and reset by software. The PLL0QEN bit can be written only if the PLL0EN is 0.<br>0: Disable the CK_PLL0Q output<br>1: Enable the CK_PLL0Q output |
| 22:16 | PLL2Q[6:0] | The PLL2Q output frequency division factor from PLL2 VCO clock<br>Set and reset by software when the PLL2 is disable. These bits used to generator PLL2Q output clock (CK_PLL2Q) from PLL2 VCO clock (CK_PLL2VCO). The CK_PLL2VCO is described in PLL2N bits in RCU_PLL2 register. |

| | | |
|---|---|---|
| | | 0000000: CK_PLL2Q = CK_PLL2VCO<br>0000001: CK_PLL2Q = CK_PLL2VCO / 2<br>0000010: CK_PLL2Q = CK_ PLL2VCO / 3.<br>0000011: CK_PLL2Q = CK_ PLL2VCO / 4<br>0000100: CK_PLL2Q = CK_ PLL2VCO / 5<br>…<br>1111111: CK_PLL2Q = CK_ PLL2VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:8 | PLL1Q[6:0] | The PLL1Q output frequency division factor from PLL1 VCO clock<br>Set and reset by software when the PLL1 is disable. These bits used to generator PLL1Q output clock (CK_PLL1Q) from PLL1 VCO clock (CK_PLL1VCO). The CK_PLL1VCO is described in PLL1N bits in RCU_PLL1 register.<br>0000000: CK_PLL1Q = CK_PLL1VCO<br>0000001: CK_ PLL1Q = CK_PLL1VCO / 2<br>0000010: CK_PLL1Q = CK_PLL1VCO / 3.<br>0000011: CK_PLL1Q = CK_PLL1VCO / 4<br>0000100: CK_PLL1Q = CK_PLL1VCO / 5<br>…<br>1111111: CK_PLL1Q = CK_PLL1VCO / 128 |
| 7 | Reserved | Must be kept at reset value. |
| 6:0 | PLL0Q[6:0] | The PLL0Q output frequency division factor from PLL0 VCO clock<br>Set and reset by software when the PLL0 is disable. These bits used to generator PLL0Q output clock (CK_PLL0Q) from PLL0 VCO clock (CK_PLL0VCO). The CK_PLL0Q is used to USBHS (48MHz), TRNG(48MHz) and SDIO. The CK_PLL0VCO is described in PLL0N bits in RCU_PLL0 register.<br>0000000: CK_PLL0Q = CK_PLL0VCO<br>0000001: CK_PLL0Q = CK_PLL0VCO / 2<br>0000010: CK_PLL0Q = CK_PLL0VCO / 3<br>0000011: CK_PLL0Q = CK_PLL0VCO / 4<br>0000100: CK_PLL0Q = CK_PLL0VCO / 5<br>…<br>1111111: CK_PLL0Q = CK_PLL0VCO / 128 |

### 6.3.32. PLL1 register (RCU_PLL1)

Address offset: 0x84

Reset value: 0x0101 2020

To configure the PLL1 clock, refer to the following formula:

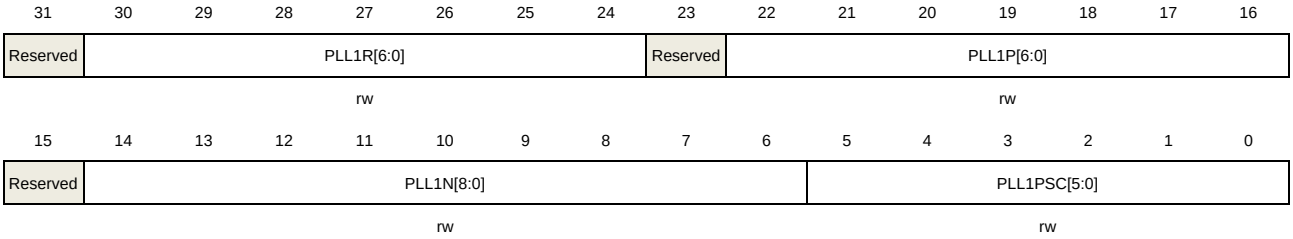
CK_PLL1VCOSRC = CK_PLL1SRC / PLL1PSC

CK_PLL1VCO = CK_PLL1VCOSRC × (PLL1N + PLL1FRAN / $2^{13}$)

CK_PLL1P = CK_ PLL1VCO / PLL1P

CK_PLL1R = CK_PLL1VCO / PLL1R

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | PLL1R[6:0] | The PLL1R output frequency division factor from PLL1 VCO clock<br>Set and reset by software when the PLL1 is disable. These bits used to generate PLL1R output clock (CK_PLL1R) from PLL1 VCO clock (CK_PLL1VCO). The CK_PLL1VCO is described in PLL1N bits in RCU_PLL1 register.<br>0000000: CK_PLL1R = CK_PLL1VCO<br>0000001: CK_PLL1R = CK_PLL1VCO / 2<br>0000010: CK_PLL1R = CK_PLL1VCO / 3.<br>0000011: CK_PLL1R = CK_PLL1VCO / 4<br>0000100: CK_PLL1R = CK_PLL1VCO / 5<br>…<br>1111111: CK_PLL1R = CK_PLL1VCO / 128 |
| 23 | Reserved | Must be kept at reset value. |
| 22:16 | PLL1P[6:0] | The PLL1P output frequency division factor from PLL1 VCO clock<br>Set and reset by software when the PLL1 is disable. These bits used to generator PLL1P output clock (CK_PLL1P) from PLL1 VCO clock (CK_PLL1VCO). The CK_PLL1VCO is described in PLL1N bits in RCU_ PLL1 register.<br>0000000: CK_PLL1P = CK_PLL1VCO<br>0000001: CK_PLL1P = CK_PLL1VCO / 2<br>0000010: CK_PLL1P = CK_PLL1VCO / 3<br>0000011: CK_PLL1P = CK_PLL1VCO / 4<br>0000100: CK_PLL1P = CK_PLL1VCO / 5<br>…<br>1111111: CK_PLL1P = CK_PLL1VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:6 | PLL1N[8:0] | The PLL1 VCO clock multiplication factor<br>Set and reset by software (only use word / half-word write) when the PLL1 is disable. These bits used to generate PLL1 VCO clock (CK_PLL1VCO) from PLL1 VCO source clock (CK_PLL1VCOSRC). The CK_PLL1VCOSRC is described in PLLPSC bits in RCU_PLL register<br>**Note:** The frequency of CK_PLL1VCO is between 150MHz to 836MHz.<br>The value of PLL1N must :<br>When PLL1 fractional latch is disabled, the value of PLL1N must satisfy: 9 ≤ PLL1N |

| | | |
|---|---|---|
| | | ≤ 512<br>When PLL1 fractional latch is enabled, the value of PLL1N must satisfy: 12 ≤ PLL1N ≤ 508<br>000000000: Reserved<br>…<br>000000111: Reserved<br>000001000: PLL1N = 9.<br>…<br>001000000: PLL1N = 65<br>001000001: PLL1N = 66<br>…<br>111111111: PLL1N = 512 |
| 5:0 | PLL1PSC[5:0] | The PLL1 VCO source clock prescaler<br>Set and reset by software when the PLL1 is disable. These bits used to generate the clock of PLL1 VCO source clock (CK_PLL1VCOSRC) from PLL1 source clock (CK_PLL1SRC) which is described in PLLSEL in RCU_PLLALL register.<br>The VCO source clock is between 1M to 16MHz.<br>000000: Reserved.<br>000001: CK_PLL1SRC<br>000010: CK_PLL1SRC / 2<br>000011: CK_PLL1SRC / 3<br>…<br>111111: CK_PLL1SRC / 63 |

## 6.3.33. PLL2 register (RCU_PLL2)

Address offset: 0x88

Reset value: 0x0101 2020

To configure the PLL2 clock, refer to the following formula:

CK_PLL2VCOSRC = CK_PLL2SRC / PLL2PSC

CK_PLL2VCO = CK_ PLL2VCOSRC × (PLL2N + PLL2FRAN / $2^{13}$)

CK_PLL2P = CK_ PLL2VCO / PLL2P

CK_PLL2R = CK_ PLL2VCO / PLL2R

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL2R[6:0] | | | | | | | Reserved | PLL2P[6:0] | | | | | | |
| | rw | | | | | | | | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL2N[8:0] | | | | | | | | | PLL2PSC[5:0] | | | | | |
| | rw | | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | PLL2R[6:0] | The PLL2R output frequency division factor from PLL2 VCO clock<br>Set and reset by software when the PLL2 is disable. These bits used to generate PLL2R output clock (CK_PLL2R) from PLL2 VCO clock (CK_PLL2VCO). The CK_PLL2R is used to generate TLI clock (≤216MHz). The CK_PLL2VCO is described in PLL2N bits in RCU_PLL2 register.<br>0000000: CK_PLL2R = CK_PLL2VCO<br>0000001: CK_PLL2R = CK_PLL2VCO / 2<br>0000010: CK_PLL2R = CK_PLL2VCO / 3<br>0000011: CK_PLL2R = CK_PLL2VCO / 4<br>0000100: CK_PLL2R = CK_PLL2VCO / 5<br>…<br>1111111: CK_PLL2R = CK_PLL2VCO / 128 |
| 23 | Reserved | Must be kept at reset value. |
| 22:16 | PLL2P[6:0] | The PLL2 P output frequency division factor from PLL2 VCO clock<br>Set and reset by software when the PLL2 is disable. These bits used to generator PLL2 P output clock (CK_PLL2P) from PLL2 VCO clock (CK_PLL2VCO). The CK_PLL2P is used to USBHS (48MHz), TRNG (48MHz), or SDIO (≤48MHz). The CK_PLL2VCO is described in PLL2N bits in RCU_PLL2 register.<br>0000000: CK_PLL2P = CK_PLL2VCO<br>0000001: CK_PLL2P = CK_PLL2VCO / 2<br>0000010: CK_PLL2P = CK_PLL2VCO / 3<br>0000011: CK_PLL2P = CK_PLL2VCO / 4<br>0000100: CK_PLL2R = CK_PLL2VCO / 5<br>…<br>1111111: CK_PLL2R = CK_PLL2VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:6 | PLL2N[8:0] | The PLL2 VCO clock multiplication factor<br>Set and reset by software (only use word / half-word write) when the PLL2 is disable. These bits used to generate PLL2 VCO clock (CK_PLL2VCO) from PLL2 VCO source clock (CK_PLL2VCOSRC). The CK_PLL2VCOSRC is described in PLLPSC bits in RCU_PLL register.<br>**Note:** The frequency of CK_PLL2VCO is between 150MHz to 836MHz<br>The value of PLL2N must :<br>When PLL2 fractional latch is disabled, the value of PLL2N must satisfy: 9 ≤ PLL2N ≤ 512<br>When PLL2 fractional latch is enabled, the value of PLL2N must satisfy: 12 ≤ PLL2N ≤ 508<br>000000000: Reserved<br>…<br>000000111: Reserved<br>000001000: PLL2N = 9. |

| | | |
|---|---|---|
| | | …<br>001000000: PLL2N = 65<br>001000001: PLL2N = 66<br>…<br>111111111: PLL2N = 512 |
| 5:0 | PLL2PSC[5:0] | The PLL2 VCO source clock prescaler<br>Set and reset by software when the PLL2 is disable. These bits used to generate the clock of PLL2 VCO source clock (CK_PLL2VCOSRC) from PLL2 source clock (CK_PLL2SRC) which is described in PLLSEL in RCU_PLLALL register.<br>The VCO source clock is between 1M to 16MHz.<br>000000: Reserved .<br>000001: CK_PLL2SRC<br>000010: CK_PLL2SRC / 2<br>000011: CK_PLL2SRC / 3<br>…<br>111111: CK_PLL2SRC / 63 |

## 6.3.34. Clock configuration register 1 (RCU_CFG1)

Address offset: 0x8C

Reset value: 0x0000 3F00

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HPDFSEL | Reserved | | | | | | TIMERSEL | USART5SEL[1:0] | | USART2SEL[1:0] | | USART1SEL[1:0] | | PLL2RDIV[1:0] | |
| rw | | | | | | | rw | rw | | rw | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PERSEL[1:0] | | CAN2SEL[1:0] | | CAN1SEL[1:0] | | CAN0SEL[1:0] | | Reserved | | RSPDIFSEL[1:0] | | Reserved | | USART0SEL[1:0] | |
| rw | | rw | | rw | | rw | | | | rw | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | HPDFSEL | HPDF clock source selection<br>Set and reset by software to control the HPDF clock source<br>0: CK_APB2 selected as HPDF source clock<br>1: CK_AHB selected as HPDF source clock |
| 30:25 | Reserved | Must be kept at reset value. |
| 24 | TIMERSEL | TIMER clock selection<br>This bit is set and reset by software. This bit defined all timer clock selection.<br>0: If APB1PSC / APB2PSC in RCU_CFG0 register is 0b0xx(CK_APBx = CK_AHB) or 0b100(CK_APBx = CK_AHB/2), the TIMER clock is equal to CK_AHB(CK_TIMERx = CK_AHB). Or else, the TIMER clock is twice the corresponding APB clock (TIMER in APB1 domain: CK_TIMERx = 2 x CK_APB1; |

| | | |
|---|---|---|
| | | TIMER in APB2 domain: CK_TIMERx = 2 x CK_APB2).<br>1: If APB1PSC / APB2PSC in RCU_CFG0 register is 0b0xx(CK_APBx = CK_AHB), 0b100(CK_APBx = CK_AHB / 2), or 0b101(CK_APBx = CK_AHB / 4), the TIMER clock is equal to CK_AHB(CK_TIMERx = CK_AHB). Or else, the TIMER clock is four times the corresponding APB clock (TIMER in APB1 domain: CK_TIMERx = 4 x CK_APB1, TIMER in APB2 domain: CK_TIMERx = 4 x CK_APB2). |
| 23:22 | USART5SEL[1:0] | USART5 clock source selection<br>Set and reset by software to control the USART5 clock source<br>00: CK_APB2 output clock selected as USART5 source clock<br>01: CK_AHB output clock selected as USART5 source clock<br>10: CK_LXTAL output clock selected as USART5 source clock<br>11: CK_IRC64MDIV output clock selected as USART5 source clock |
| 21:20 | USART2SEL[1:0] | USART2 clock source selection<br>Set and reset by software to control the USART2 clock source<br>00: CK_APB1 selected as USART2 source clock<br>01: CK_AHB selected as USART2 source clock<br>10: CK_LXTAL selected as USART2 source clock<br>11: CK_IRC64MDIV selected as USART2 source clock |
| 19:18 | USART1SEL[1:0] | USART1 clock source selection<br>Set and reset by software to control the USART1 clock source<br>00: CK_APB1 selected as USART1 source clock<br>01: CK_AHB selected as USART1 source clock<br>10: CK_LXTAL selected as USART1 source clock<br>11: CK_IRC64MDIV selected as USART1 source clock |
| 17:16 | PLL2RDIV[1:0] | The divider factor from PLL2R clock<br>These bits are set and reset by software when PLL2 is disabled. These bits used to generate clock for TLI clock.<br>00: CK_PLL2R / 2<br>01: CK_PLL2R / 4<br>10: CK_PLL2R / 8<br>11: CK_PLL2R / 16 |
| 15:14 | PERSEL[1:0] | CK_PER clock selection<br>Set and reset by software to control the CK_PER clock source<br>00: CK_IRC64MDIV selected as CK_PER source clock<br>01: CK_LPIRC4M selected as CK_PER source clock<br>10: CK_HXTAL selected as CK_PER source clock<br>11: reserved |
| 13:12 | CAN2SEL[1:0] | CAN2 clock source selection<br>Set and reset by software to control the CAN2 clock source<br>00: CK_HXTAL selected as CAN2 source clock<br>01: CK_APB2 selected as CAN2 source clock |

| | | |
|---|---|---|
| | | 10: CK_APB2 / 2 selected as CAN2 source clock<br>11: CK_IRC64MDIV selected as CAN2 source clock |
| 11:10 | CAN1SEL[1:0] | CAN1 clock selection<br>Set and reset by software to control the CAN1 clock source<br>00: CK_HXTAL selected as CAN1 source clock<br>01: CK_APB2 selected as CAN1 source clock<br>10: CK_APB2 / 2 selected as CAN1 source clock<br>11: CK_IRC64MDIV selected as CAN1 source clock |
| 9:8 | CAN0SEL[1:0] | CAN0 clock source selection<br>Set and reset by software to control the CAN0 clock source<br>00: CK_HXTAL selected as CAN0 source clock<br>01: CK_APB2 selected as CAN0 source clock<br>10: CK_APB2 / 2 selected as CAN0 source clock<br>11: CK_IRC64MDIV selected as CAN0 source clock |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | RSPDIFSEL[1:0] | RSPDIF clock source selection<br>Set and reset by software to control the RSPDIF clock source<br>00: CK_PLL0Q selected as RSPDIF source clock<br>01: CK_PLL1R selected as RSPDIF source clock<br>10: CK_PLL2R selected as RSPDIF source clock<br>11: CK_IRC64MDIV selected as RSPDIF source clock |
| 3:2 | Reserved | Must be kept at reset value. |
| 1:0 | USART0SEL[1:0] | USART0 clock source selection<br>Set and reset by software to control the USART0 clock source<br>00: CK_APB2 selected as USART0 source clock<br>01: CK_AHB selected as USART0 source clock<br>10: CK_LXTAL selected as USART0 source clock<br>11: CK_IRC64MDIV selected as USART0 source clock |

### 6.3.35. Clock configuration register 2 (RCU_CFG2)

Address offset: 0x90

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 29 28 | 27 | 26 25 24 | 23 | 22 21 20 | 19 | 18 17 16 |
|---|---|---|---|---|---|---|---|
| Reserved | SAI2B1SEL[2:0] | Reserved | SAI2B0SEL[2:0] | Reserved | SAI1SEL[2:0] | Reserved | SAI0SEL[2:0] |
| | rw | | rw | | rw | | rw |

| 15 | 14 13 12 | 11 | 10 9 8 | 7 | 6 5 4 | 3 | 2 1 0 |
|---|---|---|---|---|---|---|---|
| Reserved | CKOUT1SEL[2:0] | CKOUT1DIV[3:0] | | Reserved | CKOUT0SEL[2:0] | CKOUT0DIV[3:0] | |
| | rw | rw | | | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:28 | SAI2B1SEL[2:0] | SAI2 Block 1 clock source selection<br>Set and reset by software to control the SAI2 BLOCK 1 clock source<br>000: CK_PLL0Q selected as SAI2 BLOCK 1 source clock<br>001: CK_PLL1P selected as SAI2 BLOCK 1 source clock<br>010: CK_PLL2P selected as SAI2 BLOCK 1 source clock<br>011: I2S_CKIN selected as SAI2 BLOCK 1 source clock<br>100: CK_PER selected as SAI2 BLOCK 1 source clock<br>101: CK_RSPDIF_SYMBselected as SAI2 BLOCK 1 source clock<br>11x: reserved |
| 27 | Reserved | Must be kept at reset value. |
| 26:24 | SAI2B0SEL[2:0] | SAI2 Block 0 clock source selection<br>Set and reset by software to control the SAI2 BLOCK 0 clock source<br>000: CK_PLL0Q selected as SAI2 BLOCK 0 source clock<br>001: CK_PLL1P selected as SAI2 BLOCK 0 source clock<br>010: CK_PLL2P selected as SAI2 BLOCK 0 source clock<br>011: I2S_CKIN selected as SAI2 BLOCK 0 source clock<br>100: CK_PER selected as SAI2 BLOCK 0 source clock<br>101: CK_RSPDIF_SYMBselected as SAI2 BLOCK 0 source clock<br>11x: reserved |
| 23 | Reserved | Must be kept at reset value. |
| 22:20 | SAI1SEL[1:0] | SAI1 clock source selection<br>Set and reset by software to control the SAI1 clock source<br>000: CK_PLL0Q selected as SAI1 source clock<br>001: CK_PLL1P selected as SAI1 source clock<br>010: CK_PLL2P selected as SAI1 source clock<br>011: I2S_CKIN selected as SAI1 source clock<br>100: CK_PER selected as SAI1 source clock<br>others: Reserved |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | SAI0SEL[1:0] | SA0 and HPDF audio clock source selection<br>Set and reset by software to control the SAI0 and HPDF audio clock source<br>000: CK_PLL0Q selected as SAI0 and HPDF audio source clock<br>001: CK_PLL1P selected as SAI0 and HPDF audio source clock<br>010: CK_PLL2P selected as SAI0 and HPDF audio source clock<br>011: I2S_CKIN selected as SAI0 and HPDF audio source clock<br>100: CK_PER selected as SAI0 and HPDF audio source clock<br>others: Reserved |

| | | |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | CKOUT1SEL[2:0] | CKOUT1 clock source selection<br>Set and reset by software.<br>000: CK_SYS clock selected<br>001: CK_PLL1R clock selected<br>010: CK_HXTAL clock selected<br>011: CK_PLL0P clock selected<br>100: CK_LPIRC4M clock selected<br>101: CK_IRC32K clock selected<br>110: CK_PLL2R clock selected<br>111: reserved<br>**Note:** Configuration of this bit field may cause interference with CK_OUT1, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs. |
| 11:8 | CKOUT1DIV[3:0] | The CK_OUT1 divider which the CK_OUT1 frequency can be reduced see bits14:12 of RCU_CFG2 for CK_OUT1<br>0000: inhibit predividers<br>0001: CK_OUT1 is divided by 1<br>0010: CK_OUT1 is divided by 2<br>0011: CK_OUT1 is divided by 3<br>0100: CK_OUT1 is divided by 4<br>…<br>1111: CK_OUT1 is divided by 15<br>**Note:** Configuration of this bit field may cause interference with CK_OUT1, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs. |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | CKOUT0SEL[2:0] | CKOUT0 clock source selection<br>Set and reset by software<br>000: CK_OUT0 selects CK_IRC64MDIV<br>001: CK_OUT0 selects CK_LXTAL<br>010: CK_OUT0 selects CK_HXTAL<br>011: CK_OUT0 selects CK_PLL0P<br>100: CK_OUT0 selects CK_IRC48M<br>101: CK_OUT0 selects CK_PER<br>110: CK_OUT0 selects USBHS0 60M<br>111: CK_OUT0 selects USBHS1 60M<br>**Note:** Configuration of this bit field may cause interference with CK_OUT0, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs. |
| 3:0 | CKOUT0DIV[3:0] | The CK_OUT0 divider which the CK_OUT0 frequency can be reduced see bits 6:4 of RCU_CFG2 for CK_OUT0 |

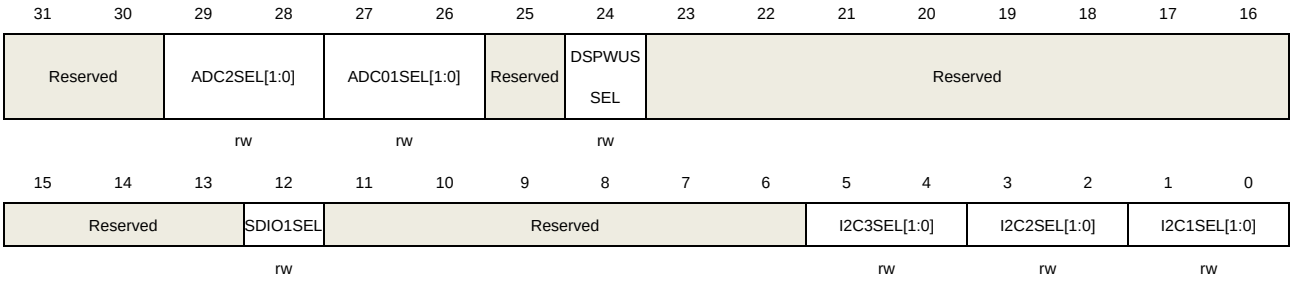
0000: inhibit predividers
0001: The CK_OUT0 is divided by 1
0010: The CK_OUT0 is divided by 2
0011: The CK_OUT0 is divided by 3
0100: The CK_OUT0 is divided by 4
...
1111: The CK_OUT0 is divided by 15

**Note:** Configuration of this bit field may cause interference with CK_OUT0, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs.

## 6.3.36. Clock configuration register 3 (RCU_CFG3)

Address offset: 0x94
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:28 | ADC2SEL[1:0] | ADC2 clock source selection<br>Set and reset by software to control the ADC2 clock source<br>00: CK_PLL1P selected as ADC2 source clock<br>01: CK_PLL2R selected as ADC2 source clock<br>10: CK_PER selected as ADC2 source clock<br>11: reserved |
| 27:26 | ADC01SEL[1:0] | ADC0 and ADC1 clock source selection<br>Set and reset by software to control the ADC0 and ADC1 clock source<br>00: CK_PLL1P selected as ADC0 and ADC1 source clock<br>01: CK_PLL2R selected as ADC0 and ADC1 source clock<br>10: CK_PER selected as ADC0 and ADC1 source clock<br>11: reserved |
| 25 | Reserved | Must be kept at reset value. |
| 24 | DSPWUSSEL | Deep-sleep wakeup system clock source selection<br>Set and reset by software to select the system wakeup clock from |

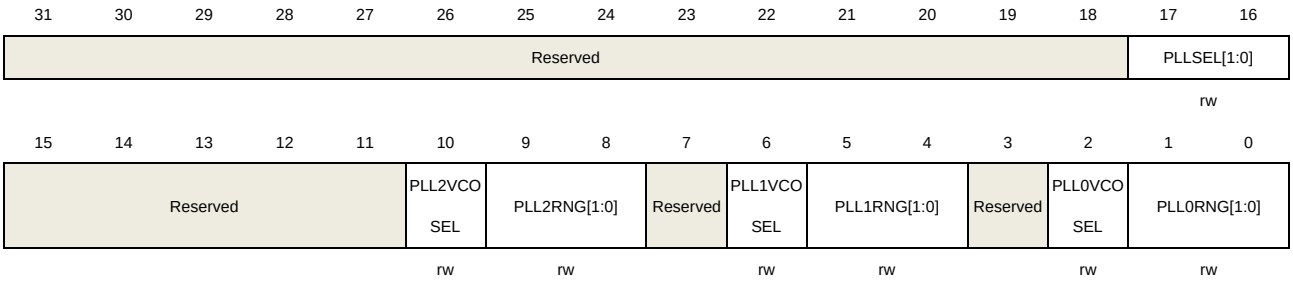
| | | |
|---|---|---|
| | | deep-sleep mode.<br>The selected clock is also used as emergency clock for the Clock stuck on HXTAL.<br>0: The CK_IRC64MDIV is selected as wake up system clock from deep-sleep mode<br>1: The CK_LPIRC4M is selected as wake up system clock from deep-sleep mode<br>**Note:** If DSPWUSSEL = '1' and peripheral clock source select CK_IRC64MDIV, when system is wakeup form deep-sleep mode by this peripheral, if peripheral wakeup function turns off, the peripheral will have no clock because of IRC64M is turn off. In this case, user needs to set IRC64MEN in RCU_CTL register to turn on IRC64M clock again. When the CKMEN bit is set and the system clock is CK_HXTAL or switched to HXTAL, this bit cannot be changed. |
| 23:13 | Reserved | Must be kept at reset value. |
| 12 | SDIO1SEL | SDIO1 clock source selection<br>Set and reset by software to control the SDIO1 clock source.<br>0: CK_PLL0Q selected as SDIO1 source clock<br>1: CK_PLL1R selected as SDIO1 source clock |
| 11:6 | Reserved | Must be kept at reset value. |
| 5:4 | I2C3SEL[1:0] | I2C3 clock source selection<br>Set and reset by software to control the I2C3 clock source.<br>00: CK_APB1 selected as I2C3 source clock<br>01: CK_PLL2R selected as I2C3 source clock<br>10: CK_IRC64MDIV selected as I2C3 source clock<br>11: CK_LPIRC4M selected as I2C3 source clock |
| 3:2 | I2C2SEL[1:0] | I2C2 clock source selection<br>Set and reset by software to control the I2C2 clock source.<br>00: CK_APB1 selected as I2C2 source clock<br>01: CK_PLL2R selected as I2C2 source clock<br>10: CK_IRC64MDIV selected as I2C2 source clock<br>11: CK_LPIRC4M selected as I2C2 source clock |
| 1:0 | I2C1SEL[1:0] | I2C1 clock source selection<br>Set and reset by software to control the I2C1 clock source.<br>00: CK_APB1 selected as I2C1 source clock<br>01: CK_PLL2R selected as I2C1 source clock<br>10: CK_IRC64MDIV selected as I2C1 source clock<br>11: CK_LPIRC4M selected as I2C1 source clock |

### 6.3.37. PLL all configuration register (RCU_PLLALL)

Address offset: 0x98

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



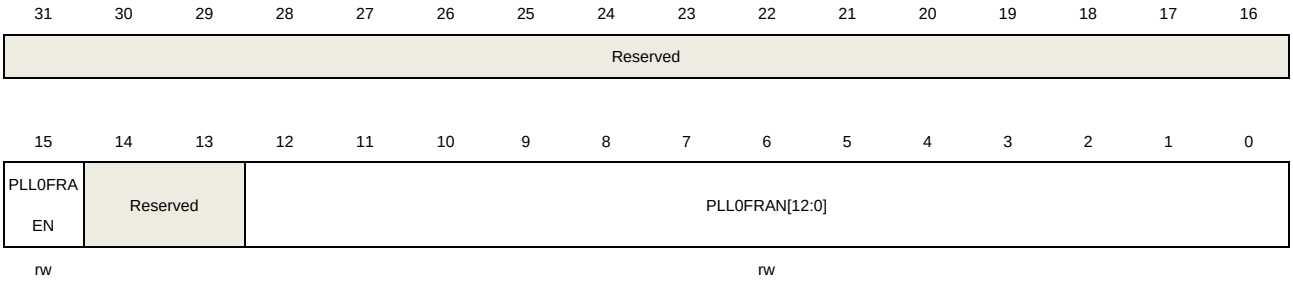

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17:16 | PLLSEL[1:0] | PLLs clock source selection<br>Set and reset by software to control the PLLs clock source.<br>00: CK_IRC64MDIV selected as source clock of PLL0, PLL1, PLL2<br>01: CK_LPIRC4M selected as source clock of PLL0, PLL1, PLL2<br>10: CK_HXTAL slected as source clock of PLL0, PLL1, PLL2<br>11: No clock selected as source clock of PLL0, PLL1, PLL2 |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | PLL2VCOSEL | PLL2 VCO selection<br>Set and reset by software when the PLL2 is disable.<br>0: select wide VCO, range: 192 - 836MHz, PLL2 input clock frequency range should be 2 -16 MHz.<br>1: select narrow VCO, range: 150 - 420MHz, PLL2 input clock frequency range should be 1 -2 MHz. |
| 9:8 | PLL2RNG[1:0] | PLL2 input clock range<br>Set and reset by software when the PLL2 is disable.<br>00: input clock frequency: 1 - 2MHz<br>01: input clock frequency: 2 - 4MHz<br>10: input clock frequency: 4 - 8MHz<br>11: input clock frequency: 8 - 16MHz |
| 7 | Reserved | Must be kept at reset value. |
| 6 | PLL1VCOSEL | PLL1 VCO selection<br>Set and reset by software when the PLL1 is disable.<br>0: select wide VCO, range: 192 - 836MHz, PLL1 input clock frequency range should be 2 -16 MHz.<br>1: select narrow VCO, range: 150 - 420MHz, PLL1 input clock frequency range |

| | | |
|---|---|---|
| | | should be 1 -2 MHz. |
| 5:4 | PLL1RNG[1:0] | PLL1 input clock range<br>Set and reset by software when the PLL1 is disable.<br>00: input clock frequency: 1 - 2MHz<br>01: input clock frequency: 2 - 4MHz<br>10: input clock frequency: 4 - 8MHz<br>11: input clock frequency: 8 - 16MHz |
| 3 | Reserved | Must be kept at reset value. |
| 2 | PLL0VCOSEL | PLL0 VCO selection<br>Set and reset by software when the PLL0 is disable.<br>0: select wide VCO, range: 192 - 836MHz, PLL0 input clock frequency range should be 2 -16 MHz.<br>1: select narrow VCO, range: 150 - 420MHz, PLL0 input clock frequency range should be 1 -2 MHz. |
| 1:0 | PLL0RNG[1:0] | PLL0 input clock range<br>Set and reset by software when the PLL0 is disable..<br>00: input clock frequency: 1 - 2MHz<br>01: input clock frequency: 2 - 4MHz<br>10: input clock frequency: 4 - 8MHz<br>11: input clock frequency: 8 - 16MHz |

### 6.3.38. PLL0 fraction configuration register (RCU_PLL0FRA)

Address offset: 0x9C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | PLL0FRAEN | PLL0 fractional latch enable.<br>These bits are set and reset by software. These bits can lock the PLL0FRAN value into a Sigma-Delta modulator .When PLL0FRAEN bit switchs from "0" to "1", the PLL0FRAN value will transfer to modulator. |

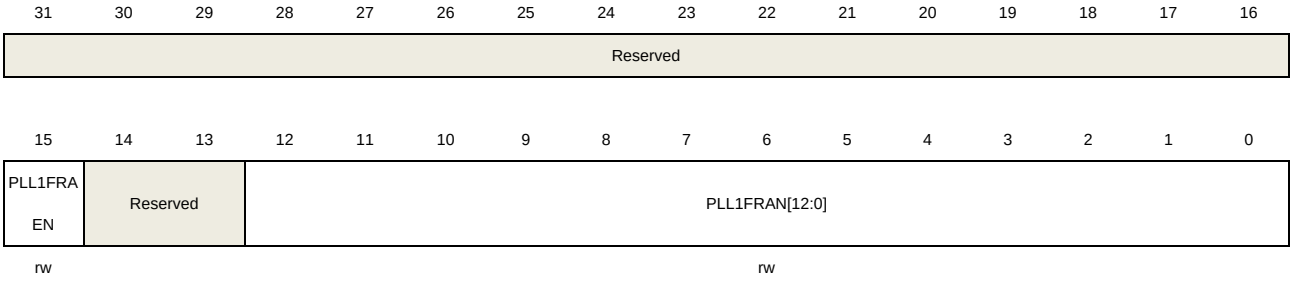
| | | |
|---|---|---|
| 14:13 | Reserved | Must be kept at reset value. |
| 12:0 | PLL0FRAN[12:0] | Fractional part of the multiplication factor for PLL0 VCO<br>These bits are set and reset by software. These bits can control the fractional part of the multiplication for PLL0 VCO. This bit field can be modified dynamically to fine-tune the PLL0 VCO.<br>These bits must configure the PLL0 VCO out frequency to the following range: When PLL0VCOSEL = 0, the range is 192MHz to 836MHz; When PLL0VCOSEL = 1, the range is 150MHz to 420MHz. |

## 6.3.39. PLL1 fraction configuration register (RCU_PLL1FRA)

Address offset: 0xA0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



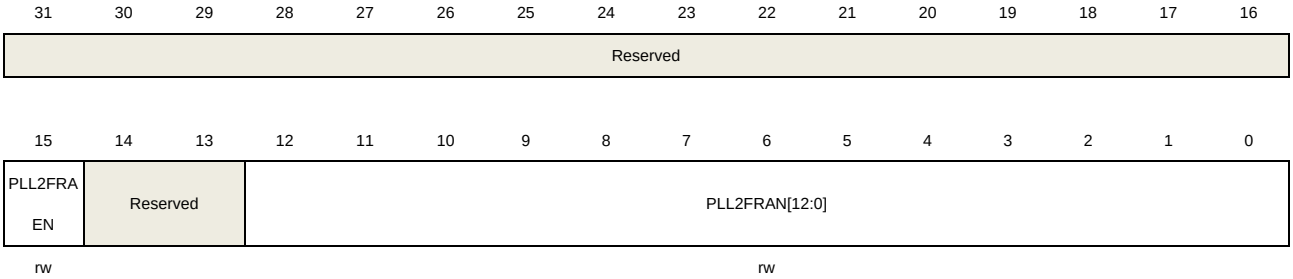

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | PLL1FRAEN | PLL1 fractional latch enable.<br>These bits are set and reset by software. These bits can lock the PLL1FRAN value into a Sigma-Delta modulator .When PLL1FRAEN bit switchs from "0" to "1", the PLL1FRAN value will transfer to modulator. |
| 14:13 | Reserved | Must be kept at reset value. |
| 12:0 | PLL1FRAN [12:0] | Fractional part of the multiplication factor for PLL1 VCO<br>These bits are set and reset by software. These bits can control the fractional part of the multiplication for PLL1 VCO. This bit field can be modified dynamically to fine-tune the PLL1 VCO.<br>These bits must configure the PLL1 VCO out frequency to the following range: When PLL1VCOSEL = 0, the range is 192MHz to 836MHz; When PLL1VCOSEL = 1, the range is 150MHz to 420MHz. |

## 6.3.40. PLL2 fraction configuration register (RCU_PLL2FRA)

Address offset: 0xA4

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



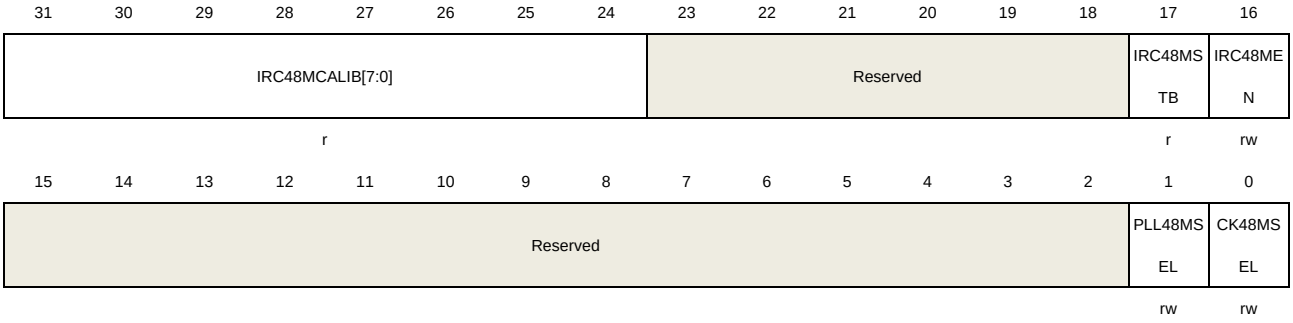

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | PLL2FRAEN | PLL2 fractional latch enable.<br>These bits are set and reset by software. These bits can lock the PLL2FRAN value into a Sigma-Delta modulator .When PLL2FRAEN bit switchs from "0" to "1", the PLL2FRAN value will transfer to modulator. |
| 14:13 | Reserved | Must be kept at reset value. |
| 12:0 | PLL2FRAN[12:0] | Fractional part of the multiplication factor for PLL2 VCO<br>These bits are set and reset by software. These bits can control the fractional part of the multiplication for PLL2 VCO. This bit field can be modified dynamically to fine-tune the PLL2 VCO.<br>These bits must configure the PLL2 VCO out frequency to the following range: When PLL2VCOSEL = 0, the range is 192MHz to 836MHz; When PLL2VCOSEL = 1, the range is 150MHz to 420MHz. |

## 6.3.41. Additional clock control register 0 (RCU_ADDCTL0)

Address offset: 0xC0

Reset value: 0x8000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | IRC48MCALIB [7:0] | Internal 48MHz RC oscillator calibration value register<br>These bits are load automatically at power on. |

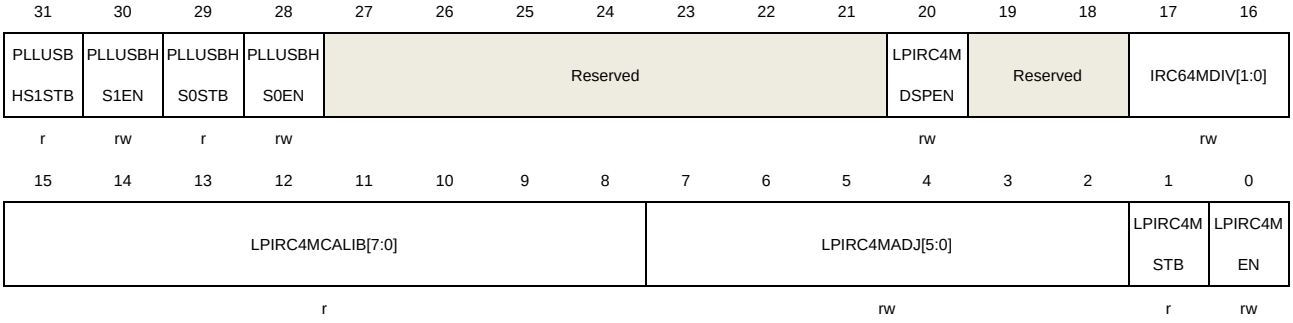
| | | |
|---|---|---|
| 23:18 | Reserved | Must be kept at reset value. |
| 17 | IRC48MSTB | Internal 48MHz RC oscillator clock stabilization Flag<br>Set by hardware to indicate if the IRC48M oscillator is stable and ready for use.<br>0: IRC48M is not stable<br>1: IRC48M is stable |
| 16 | IRC48MEN | Internal 48MHz RC oscillator enable<br>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: IRC48M disable<br>1: IRC48M enable |
| 15:2 | Reserved | Must be kept at reset value. |
| 1 | PLL48MSEL | PLL48M clock selection<br>Set and reset by software. This bit used to generate PLL48M clock which select CK_PLL0Q or CK_PLL2P clock.<br>0: Select CK_PLL0Q clock<br>1: Select CK_PLL2P clock |
| 0 | CK48MSEL | 48MHz clock selection<br>Set and reset by software. This bit used to generate CK48M clock which select IRC48M clock or PLL48M clock. The CK48M clock used for TRNG/USBHS. The PLL48M clock refer to PLL48MSEL bit in RCU_ADDCTL register.<br>0: Don't select IRC48M clock (use CK_PLL0Q clock or CK_PLL2P clock select by PLL48MSEL)<br>1: Select IRC48M clock |

## 6.3.42. Additional clock control register 1(RCU_ADDCTL1)

Address offset: 0xC4

Reset value: 0x0000 7080

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLLUSBHS1STB | PLLUSBHS1 clock stabilization flag<br>Set by hardware to indicate if the PLLUSBHS1 clock is stable and ready for use |

| | | |
|---|---|---|
| | | 0: PLLUSBHS1 clock is not stable<br>1: PLLUSBHS1 clock is stabilized |
| 30 | PLLUSBHS1EN | PLLUSBHS1 clock enable<br>Set and reset by software. Reset by hardware when enter deep-sleep or standby mode.<br>0: Disabled PLLUSBHS1 clock<br>1: Enabled PLLUSBHS1 clock |
| 29 | PLLUSBHS0STB | PLLUSBHS0 clock stabilization flag<br>Set by hardware to indicate if the PLLUSBHS0 clock is stable and ready for use<br>0: PLLUSBHS0 clock is not stable<br>1: PLLUSBHS0 clock is stabilized |
| 28 | PLLUSBHS0EN | PLLUSBHS0 clock enable<br>Set and reset by software. Reset by hardware when enter deep-sleep or standby mode.<br>0: Disabled PLLUSBHS0 clock<br>1: Enabled PLLUSBHS0 clock |
| 27:21 | Reserved | Must be kept at reset value. |
| 20 | LPIRC4MDSPEN | LPIRC4M clock enable in deepsleep mode<br>Set and reset by software. LPIRC4M can be forced on even in deepsleep mode to quickly be used as a kernel clock for some peripherals. This bit has no effect on the value of LPIRC4MEN.<br>0: Has no impact on LPIRC4M<br>1: Force LPIRC4M to run even in deepsleep mode |
| 19:18 | Reserved | Must be kept at reset value. |
| 17:16 | IRC64MDIV[1:0] | IRC64M clock divider<br>Set and reset by software. It cannot be written when system clock select CK_IRC64MDIV or IRC64MEN is set.<br>00: CK_IRC64MDIV = CK_IRC64M / 1<br>01: CK_IRC64MDIV = CK_IRC64M / 2<br>10: CK_IRC64MDIV = CK_IRC64M / 4<br>11: CK_IRC64MDIV = CK_IRC64M / 8 |
| 15:8 | LPIRC4MCALIB[7:0] | LPIRC4M calibration value<br>These bits are load automatically at power on.The calibration signal step is 0.4%. |
| 7:2 | LPIRC4MADJ[5:0] | LPIRC4M frequency clock trim adjust value<br>These bits are set by software. The trimming value is these bits (LPIRC4MADJ[) added to the LPIRC4MCALIB[7:0] bits. The trimming value should trim the LPIRC4MCALIB to 4 MHz ± 1%. |
| 1 | LPIRC4MSTB | LPIRC4M clock stabilization flag<br>Set by hardware to indicate if the LPIRC4M oscillator is stable and ready for use. |

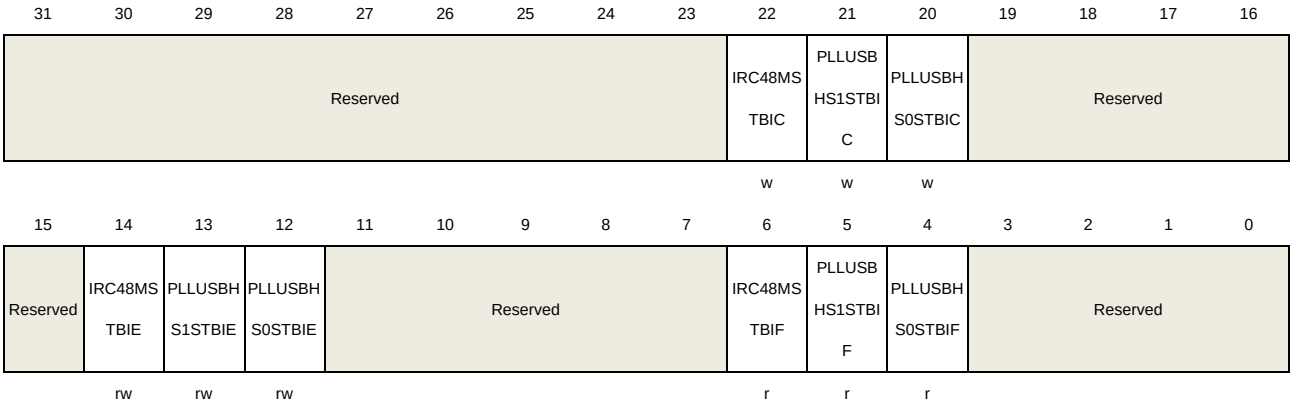
| | | |
|---|---|---|
| | | 0: LPIRC4M RC oscillator is not stable<br>1: LPIRC4M RC oscillator is stabilized |
| 0 | LPIRC4MEN | LPIRC4M clock enable<br>Set and reset by software. This bit cannot be reset if the LPIRC4M clock is used as the system clock. If DSPWUSSEL = 1, set by hardware when leaving Deep-sleep mode.<br>0: LPIRC4M RC oscillator disabled<br>1: LPIRC4M RC oscillator enabled |

## 6.3.43. Additional clock interrupt register (RCU_ADDINT)

Address offset: 0xCC

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | IRC48MSTBIC | Internal 48 MHz RC oscillator Stabilization interrupt clear<br>Write 1 by software to reset the IRC48MSTBIF flag.<br>0: Not reset IRC48MSTBIF flag<br>1: Reset IRC48MSTBIF flag |
| 21 | PLLUSBHS1STBIC | Internal PLL of USBHS1 Stabilization interrupt clear<br>Write 1 by software to reset the PLLUSBHS1STBIF flag.<br>0: Not reset PLLUSBHS1STBIF flag<br>1: Reset PLLUSBHS1STBIF flag |
| 20 | PLLUSBHS0STBIC | Internal PLL of USBHS0 Stabilization interrupt clear<br>Write 1 by software to reset the PLLUSBHS0STBIF flag.<br>0: Not reset PLLUSBHS0STBIF flag<br>1: Reset PLLUSBHS0STBIF flag |
| 19:15 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 14 | IRC48MSTBIE | Internal 48 MHz RC oscillator Stabilization interrupt enable<br>Set and reset by software to enable/disable the IRC48M stabilization interrupt<br>0: Disable the IRC48M stabilization interrupt<br>1: Enable the IRC48M stabilization interrupt |
| 13 | PLLUSBHS1STBIE | Internal PLL of USBHS1 Stabilization interrupt enable<br>Set and reset by software to enable/disable the USBHS1 PLL stabilization interrupt<br>0: Disable the USBHS1 PLL stabilization interrupt<br>1: Enable the USBHS1 PLL stabilization interrupt |
| 12 | PLLUSBHS0STBIE | Internal PLL of USBHS0 Stabilization interrupt enable<br>Set and reset by software to enable/disable the USBHS0 PLL stabilization interrupt<br>0: Disable the USBHS0 PLL stabilization interrupt<br>1: Enable the USBHS0 PLL stabilization interrupt |
| 11:7 | Reserved | Must be kept at reset value. |
| 6 | IRC48MSTBIF | IRC48M stabilization interrupt flag<br>Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set.<br>Reset by software when setting the IRC48MSTBIC bit.<br>0: No IRC48M stabilization interrupt generated<br>1: IRC48M stabilization interrupt generated |
| 5 | PLLUSBHS1STBIF | Internal PLL of USBHS1 stabilization interrupt flag<br>Set by hardware when the USBHS1 PLL clock is stable and the PLLUSBHS1STBIE bit is set.<br>Reset by software when setting the PLLUSBHS1STBIC bit.<br>0: No USBHS1 PLL clock stabilization interrupt generated<br>1: USBHS1 PLL clock stabilization interrupt generated |
| 4 | PLLUSBHS0STBIF | Internal PLL of USBHS0 stabilization interrupt flag<br>Set by hardware when the USBHS0 PLL clock is stable and the PLLUSBHS0STBIE bit is set.<br>Reset by software when setting the PLLUSBHS0STBIC bit.<br>0: No USBHS0 PLL clock stabilization interrupt generated<br>1: USBHS0 PLL clock stabilization interrupt generated |
| 3:0 | Reserved | Must be kept at reset value. |

### 6.3.44. Clock configuration register 4 (RCU_CFG4)

Address offset: 0xD0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | EXMCSEL[1:0] | | Reserved | | | | | | | SDIO0SEL |
| | | | | | | rw | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9:8 | EXMCSEL[1:0] | EXMC clock source selection<br>Set and reset by software to control the EXMC clock source<br>00: CK_AHB selected as EXMC source clock<br>01: CK_PLL0Q selected as EXMC source clock<br>10: CK_PLL1R selected as EXMC source clock<br>11: CK_PER selected as EXMC source clock |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | SDIO0SEL | SDIO0 clock source selection<br>Set and reset by software to control the SDIO0 clock source.<br>0: CK_PLL0Q selected as SDIO0 source clock<br>1: CK_PLL1R selected as SDIO0 source clock |

## 6.3.45. USB clock control register (RCU_USBCLKCTL)

Address offset: 0xD4

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | USBHS1PSC[2:0] | | | USBHS0PSC[2:0] | | |
| | | | | | | | | | | rw | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USBHS148MSEL[1:0] | | USBHS1SWEN | PLLUSBHS1PRESEL | Reserved | USBHS1SEL | Reserved | | USBHS048MSEL[1:0] | | USBHS0SWEN | PLLUSBHS0PRESEL | Reserved | USBHS0SEL | Reserved |
| | rw | | rw | rw | | rw | | | rw | | rw | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:19 | USBHS1PSC[2:0] | USBHS1 clock prescaler selection<br>Set and reset by software to control the USBHS1 clock prescaler value. The USBHS1 clock must be 48MHz. These bits can't be reset if the USBHS1 clock is enabled.<br>000: CK_USBHS1 = CK_PLL1Q / 1 |

001: CK_USBHS1 = CK_PLL1Q / 2
010: CK_USBHS1 = CK_PLL1Q / 3
011: CK_USBHS1 = CK_PLL1Q / 4
100: CK_USBHS1 = CK_PLL1Q / 5
101: CK_USBHS1 = CK_PLL1Q / 6
110 :CK_USBHS1 = CK_PLL1Q / 7
111 :CK_USBHS1 = CK_PLL1Q / 8

| | | |
|---|---|---|
| 18:16 | USBHS0PSC[2:0] | USBHS0 clock prescaler selection<br>Set and reset by software to control the USBHS0 clock prescaler value. The USBHS0 clock must be 48MHz. These bits can't be reset if the USBHS0 clock is enabled.<br>000: CK_USBHS0 = CK_PLL1Q / 1<br>001: CK_USBHS0 = CK_PLL1Q / 2<br>010: CK_USBHS0 = CK_PLL1Q / 3<br>011: CK_USBHS0 = CK_PLL1Q / 4<br>100: CK_USBHS0 = CK_PLL1Q / 5<br>101: CK_USBHS0 = CK_PLL1Q / 6<br>110 :CK_USBHS0 = CK_PLL1Q / 7<br>111 :CK_USBHS0 = CK_PLL1Q / 8 |
| 15 | Reserved | Must be kept at reset value. |
| 14:13 | USBHS148MSEL[1:0] | USBHS1 48M clock source selection<br>Set and reset by software.<br>00: CK_PLL0R selected as USBHS1 48M source clock<br>01: CK_PLLUSBHS1/USBHS1DV output clock selected as USBHS1 48M source clock<br>10: CK_PLL1Q/USBHS1PSC output selected as USBHS1 48M source clock<br>11: CK_IRC48M selected as USBHS1 48M source clock |
| 12 | USBHS1SWEN | USBHS1 clock source selection enable<br>0: Hardware switch USBHS1 clock by USBHS1 module<br>1: Use USBHS1SW to switch USBHS1 clock |
| 11 | PLLUSBHS1PRESEL | PLLUSBHS1 clock source preselection<br>Set and reset by software.<br>0: CK_HATAL selected as PLLUSBHS1 source clock<br>1: CK_IRC48M selected as PLLUSBHS1 source clock |
| 10 | Reserved | Must be kept at reset value. |
| 9 | USBHS1SEL | USBHS1 clock source selection<br>Set and reset by software.<br>0: 48M selected as USBHS1 source clock<br>1: 60M selected as USBHS1 source clock |
| 8:7 | Reserved | Must be kept at reset value. |

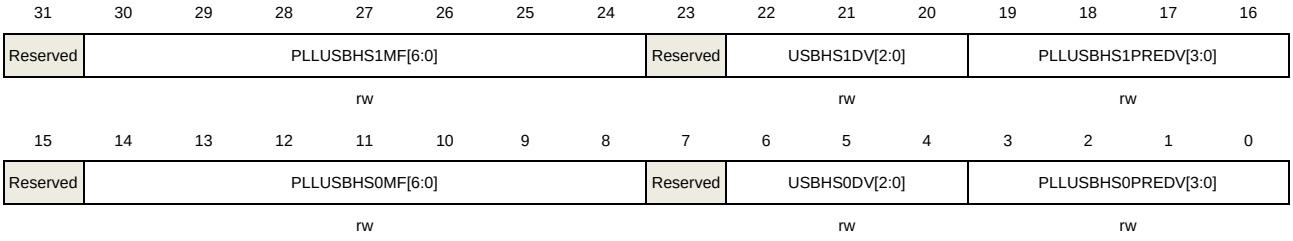
| | | |
|---|---|---|
| 6:5 | USBHS048MSEL[1:0] | USBHS0 48M clock source selection<br>Set and reset by software.<br>00: CK_PLL0R selected as USBHS0 48M source clock<br>01: CK_PLLUSBHS0/USBHS0DV selected as USBHS0 48M source clock<br>10: CK_PLL1Q/USBHS0PSC selected as USBHS0 48M source clock<br>11: CK_IRC48M selected as USBHS0 48M source clock |
| 4 | USBHS0SWEN | USBHS0 clock source selection enable<br>0: Hardware switch USBHS0 clock by USBHS0 module<br>1: Use USBHS0SEL to switch USBHS0 clock |
| 3 | PLLUSBHS0PRESEL | PLLUSBHS0 clock source preselection<br>Set and reset by software.<br>0: CK_HATAL selected as PLLUSBHS0 source clock<br>1: CK_IRC48M selected as PLLUSBHS0 source clock |
| 2 | Reserved | Must be kept at reset value. |
| 1 | USBHS0SEL | USBHS0 clock source selection<br>Set and reset by software.<br>0: 48M selected as USBHS0 source clock<br>1: 60M selected as USBHS0 source clock |
| 0 | Reserved | Must be kept at reset value. |

## 6.3.46. PLLUSB configuration register (RCU_PLLUSBCFG)

Address offset: 0xD8

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | PLLUSBHS1MF[6:0] | The PLLUSBHS1 clock multiplication factor<br>0000000: Reserved<br>0000001: Reserved<br>…<br>0001111: Reserved<br>0010000: PLLUSBHS1MF input source clock multiplied by 16 |

| | | |
|---|---|---|
| | | 0010000: PLLUSBHS1MF input source clock multiplied by 17<br>0010010: PLLUSBHS1MF input source clock multiplied by 18<br>0010011: PLLUSBHS1MF input source clock multiplied by 19<br>…<br>1111111: PLLUSBHS1MF input source clock multiplied by 127<br>**Note:** The frequency of PLLUSBHS1 output clock is no more than 480MHz |
| 23 | Reserved | Must be kept at reset value. |
| 22:20 | USBHS1DV[2:0] | USBHS1 clock divider factor<br>These bits are set and reset by software.<br>000: USBHS1DV input source clock divided by 2<br>001: USBHS1DV input source clock divided by 4<br>010: USBHS1DV input source clock divided by 6<br>…<br>111: USBHS1DV input source clock divided by 16 |
| 19:16 | PLLUSBHS1PREDV[3:0] | PLLUSBHS1PREDV clock divide factor<br>These bits are set and reset by software.<br>0000: Reserved<br>0001: PLLUSBHS1PREDV input source clock divided by 1<br>0010: PLLUSBHS1PREDV input source clock divided by / 2<br>…<br>1111: PLLUSBHS1PREDV input source clock divided by / 15 |
| 15 | Reserved | Must be kept at reset value. |
| 14:8 | PLLUSBHS0MF[6:0] | The PLLUSBHS0 clock multiplication factor<br>0000000: Reserved<br>0000001: Reserved<br>…<br>0001111: Reserved<br>0010000: PLLUSBHS0MF input source clock multiplied by 16<br>0010000: PLLUSBHS0MF input source clock multiplied by 17<br>0010010: PLLUSBHS0MF input source clock multiplied by 18<br>0010011: PLLUSBHS0MF input source clock multiplied by 19<br>…<br>1111111: PLLUSBHS0MF input source clock multiplied by 127<br>**Note:** The frequency of PLLUSBHS0 output clock is no more than 480MHz |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | USBHS0DV[2:0] | USBHS0 clock divider factor<br>These bits are set and reset by software.<br>000: USBHS0DV input source clock divided by 2<br>001: USBHS0DV input source clock divided by 4<br>010: USBHS0DV input source clock divided by 6<br>… |

| | | |
|---|---|---|
| | | 111: USBHS0DV input source clock divided by 16 |
| 3:0 | PLLUSBHS0PREDV[3:0] | PLLUSBHS0PREDV clock divide factor<br>These bits are set and reset by software.<br>0000: Reserved<br>0001: PLLUSBHS0PREDV input source clock divided by 1<br>0010: PLLUSBHS0PREDV input source clock divided by / 2<br>…<br>1111: PLLUSBHS0PREDV input source clock divided by / 15 |

## 6.3.47. APB2 additional reset register (RCU_ADDAPB2RST)

Address offset: 0xE0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | CAN2RST | CAN1RST | CAN0RST |
| | | | | | | | | | | | | | rw | rw | rw |

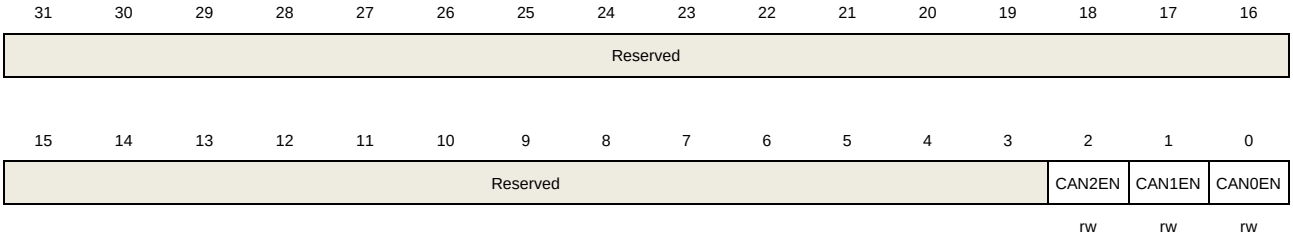
| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | CAN2RST | CAN2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN2 unit |
| 1 | CAN1RST | CAN1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN1 |
| 0 | CAN0RST | CAN0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN0 |

## 6.3.48. APB2 additional enable register (RCU_ADDAPB2EN)

Address offset: 0xE4

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



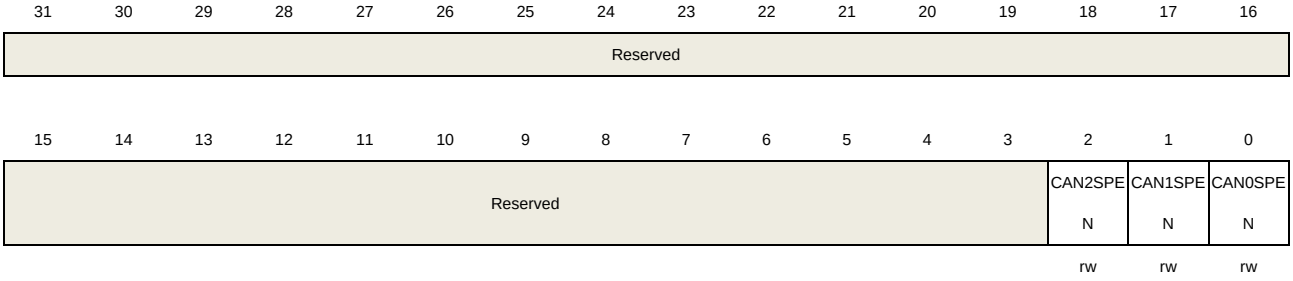

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | CAN2EN | CAN2 clock enable<br>This bit is set and reset by software.<br>0: Disable CAN2 clock<br>1: Enable CAN2 clock |
| 1 | CAN1EN | CAN1 clock enable<br>This bit is set and reset by software.<br>0: Disable CAN1 clock<br>1: Enable CAN1 clock |
| 0 | CAN0EN | CAN0 clock enable<br>This bit is set and reset by software.<br>0: Disable CAN0 clock<br>1: Enable CAN0 clock |

## 6.3.49. APB2 additional sleep enable register (RCU_ADDAPB2SPEN)

Address offset: 0xE8

Reset value: 0x0000 0007

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 30:3 | Reserved | Must be kept at reset value. |
| 2 | CAN2SPEN | CAN2 clock enable in sleep mode<br>This bit is set and reset by software.<br>0: Disable CAN2 clock in sleep mode |

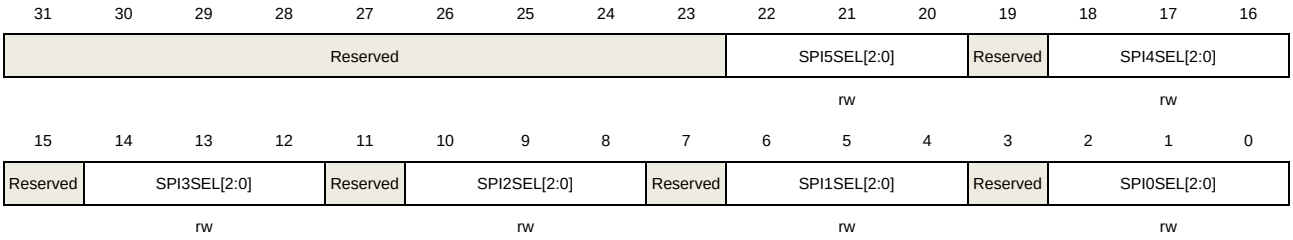
| | | |
|---|---|---|
| | | 1: Enable CAN2 clock in sleep mode |
| 1 | CAN1SPEN | CAN1 clock enable in sleep mode<br>This bit is set and reset by software.<br>0: Disable CAN1 clock in sleep mode<br>1: Enable CAN1 clock in sleep mode |
| 0 | CAN0SPEN | CAN0 clock enable in sleep mode<br>This bit is set and reset by software.<br>0: Disable CAN0 clock in sleep mode<br>1: Enable CAN0 clock in sleep mode |

### 6.3.50. Clock configuration register 5 (RCU_CFG5)

Address offset: 0xF0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22:20 | SPI5SEL[2:0] | SPI5 / I2S5 clock source selection<br>Set and reset by software to control the SPI5 / I2S5 clock source<br>000: CK_APB2 selected as SPI5 / I2S5 source clock<br>001: CK_PLL1Q selected as SPI5 / I2S5 source clock<br>010: CK_PLL2Q selected as SPI5 / I2S5 source clock<br>011: CK_IRC64MDIV selected as SPI5 / I2S5 source clock<br>100: CK_LPIRC4M selected as SPI5 / I2S5 source clock<br>101: CK_HXTAL selected as SPI5 / I2S5 source clock<br>110: I2S_CKIN selected as SPI5 / I2S5 source clock<br>111: reserved |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | SPI4SEL[2:0] | SPI4 clock source selection<br>Set and reset by software to control the SPI4 clock source<br>000: CK_APB2 selected as SPI4 source clock<br>001: CK_PLL1Q selected as SPI4 source clock<br>010: CK_PLL2Q selected as SPI4 source clock<br>011: CK_IRC64MDIV selected as SPI4 source clock |

| | | |
|---|---|---|
| | | 100: CK_LPIRC4M selected as SPI4 source clock<br>101: CK_HXTAL selected as SPI4 source clock<br>others: reserved |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | SPI3SEL[2:0] | SPI3 clock source selection<br>Set and reset by software to control the SPI3 clock source<br>000: CK_APB2 selected as SPI3 source clock<br>001: CK_PLL1Q selected as SPI3 source clock<br>010: CK_PLL2Q selected as SPI3 source clock<br>011: CK_IRC64MDIV selected as SPI3 source clock<br>100: CK_LPIRC4M selected as SPI3 source clock<br>101: CK_HXTAL selected as SPI3 source clock<br>others: reserved |
| 11 | Reserved | Must be kept at reset value. |
| 10:8 | SPI2SEL[2:0] | SPI2 / I2S2 clock source selection<br>Set and reset by software to control the SPI2 / I2S2 clock source<br>000: CK_PLL0Q selected as SPI2 / I2S2 source clock<br>001: CK_PLL1P selected as SPI2 / I2S2 source clock<br>010: CK_PLL2P selected as SPI2 / I2S2 source clock<br>011: I2S_CKIN selected as SPI2 / I2S2 source clock<br>100: CK_PER selected as SPI2 / I2S2 source clock<br>others: reserved |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | SPI1SEL[2:0] | SPI1 / I2S1 clock source selection<br>Set and reset by software to control the SPI1 / I2S1 clock source<br>000: CK_PLL0Q selected as SPI1 / I2S1 source clock<br>001: CK_PLL1P selected as SPI1 / I2S1 source clock<br>010: CK_PLL2P selected as SPI1 / I2S1 source clock<br>011: I2S_CKIN selected as SPI1 / I2S1 source clock<br>100: CK_PER selected as SPI1 / I2S1 source clock<br>others: reserved |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | SPI0SEL[2:0] | SPI0 / I2S0 clock source selection<br>Set and reset by software to control the SPI0 / I2S0 clock source<br>000: CK_PLL0Q selected as SPI0 / I2S0 source clock<br>001: CK_PLL1P selected as SPI0 / I2S0 source clock<br>010: CK_PLL2P selected as SPI0 / I2S0 source clock<br>011: I2S_CKIN selected as SPI0 / I2S0 source clock<br>100: CK_PER selected as SPI0 / I2S0 source clock<br>others: reserved |

## 6.4. Register definition(For GD32H75E)

RCU base address: 0x5802 4400

### 6.4.1. Control register (RCU_CTL)

Address offset: 0x00
Reset value: 0xC000 8040

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC64MSTB | IRC64MEN | PLL2STB | PLL2EN | PLL1STB | PLL1EN | PLL0STB | PLL0EN | Reserved | | | | CKMEN | HXTALBPS | HXTALSTB | HXTALEN |
| r | rw | r | rw | r | rw | r | rw | | | | | rw | rw | r | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IRC64MCALIB[8:0] | | | | | | | | IRC64MADJ[6:0] | | | | | | | |
| r | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | IRC64MSTB | IRC64M internal 64MHz RC oscillator stabilization flag<br>Set by hardware to indicate if the IRC64M oscillator is stable and ready for use.<br>0: IRC64M oscillator is not stable<br>1: IRC64M oscillator is stable |
| 30 | IRC64MEN | Internal 64MHz RC oscillator enable<br>Set and reset by software. This bit cannot be reset if the IRC64M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when CKMEN is set.<br>0: Internal 64 MHz RC oscillator disabled<br>1: Internal 64 MHz RC oscillator enabled |
| 29 | PLL2STB | PLL2 clock stabilization flag<br>Set by hardware to indicate if the PLL2 output clock is stable and ready for use.<br>0: PLL2 is not stable<br>1: PLL2 is stable |
| 28 | PLL2EN | PLL2 enable<br>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL2 is switched off<br>1: PLL2 is switched on |
| 27 | PLL1STB | PLL1 clock stabilization flag<br>Set by hardware to indicate if the PLL1 output clock is stable and ready for use.<br>0: PLL1 is not stable<br>1: PLL1 is stable |

| | | |
|---|---|---|
| 26 | PLL1EN | PLL1 enable<br>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL1 is switched off<br>1: PLL1 is switched on |
| 25 | PLL0STB | PLL0 clock stabilization flag<br>Set by hardware to indicate if the PLL0 output clock is stable and ready for use.<br>0: PLL0 is not stable<br>1: PLL0 is stable |
| 24 | PLL0EN | PLL0 enable<br>Set and reset by software. This bit cannot be reset if the PLL0 clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: PLL0 is switched off<br>1: PLL0 is switched on |
| 23:20 | Reserved | Must be kept at reset value. |
| 19 | CKMEN | HXTAL clock monitor enable<br>0: Disable the High speed 4 ~ 50 MHz crystal oscillator (HXTAL) clock monitor<br>1: Enable the High speed 4 ~ 50 MHz crystal oscillator (HXTAL) clock monitor<br>When the hardware detects that the HXTAL clock is stuck at a low or high state, the internal hardware will switch the system clock to be the internal high speed IRC64M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software.<br>**Note:** When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC64M internal RC oscillator regardless of the control bit, IRC64MEN, state. |
| 18 | HXTALBPS | High speed crystal oscillator (HXTAL) clock bypass mode enable<br>The HXTALBPS bit can be written only if the HXTALEN is 0.<br>0: Disable the HXTAL Bypass mode<br>1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock. |
| 17 | HXTALSTB | High speed crystal oscillator (HXTAL) clock stabilization flag<br>Set by hardware to indicate if the HXTAL oscillator is stable and ready for use.<br>0: HXTAL oscillator is not stable<br>1: HXTAL oscillator is stable |
| 16 | HXTALEN | High speed crystal oscillator (HXTAL) enable<br>Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL0 input clock when PLL0P clock is selected to the system clock. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: High speed 4 ~ 50 MHz crystal oscillator disabled<br>1: High speed 4 ~ 50 MHz crystal oscillator enabled |

| | | |
|---|---|---|
| 15:7 | IRC64MCALIB[8:0] | Internal 64MHz RC Oscillator calibration value<br>These bits are load automatically at power on. |
| 6:0 | IRC64MADJ[6:0] | Internal 64MHz RC Oscillator clock trim adjust value<br>These bits are set by software. The trimming value is these bits IRC64MADJ[6:0] added to the IRC64MCALIB[8:0] bits. The trimming value should trim the IRC64M to 64 MHz ± 1%. |

### 6.4.2. PLL0 register (RCU_PLL0)

Address offset: 0x04

Reset value: 0x0100 2020

To configure the PLL0 clock, refer to the following formula:

CK_PLL0VCOSRC = CK_PLL0SRC / PLL0PSC

CK_PLL0VCO = CK_PLL0VCOSRC × (PLL0N + PLL0FRAN / $2^{13}$)

CK_PLL0R = CK_PLL0VCO / PLL0R

CK_PLL0P = CK_PLL0VCO / PLL0P

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLLSTBSRC | PLL0R[6:0] | | | | | | | Reserved | PLL0P[6:0] | | | | | | |
| w | rw | | | | | | | | rw | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | PLL0N[8:0] | | | | | | | | | PLL0PSC[5:0] | | | | | |
| | rw | | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLLSTBSRC | PLLs stabilization signal sources.<br>0: Analog logic<br>1: Digital logic |
| 30:24 | PLL0R[6:0] | The PLL0R output frequency division factor from PLL0 VCO clock<br>Set and reset by software when the PLL0 is disable. These bits used to generator PLL0R output clock (CK_PLL0R) from PLL0 VCO clock (CK_PLL0VCO). The CK_PLL0VCO is described in PLL0N bits in RCU_PLL0 register.<br>0000000: CK_PLL0R = CK_PLL0VCO<br>0000001: CK_PLL0R = CK_PLL0VCO / 2<br>0000010: CK_PLL0R = CK_PLL0VCO / 3.<br>0000011: CK_PLL0R = CK_PLL0VCO / 4<br>0000100: CK_PLL0R = CK_PLL0VCO / 5<br>…<br>1111111: CK_PLL0R = CK_PLL0VCO / 128 |
| 23 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 22:16 | PLL0P[6:0] | The PLL0P output frequency division factor from PLL0 VCO clock<br>Set and reset by software when the PLL0 is disable. These bits used to generator PLL0P output clock (CK_PLL0P) from PLL0 VCO clock (CK_PLL0VCO). The CK_PLL0P is used to system clock (no more than 600MHz). The CK_PLL0VCO is described in PLL0N bits in RCU_PLL0 register.<br>0000000: CK_PLL0P = CK_PLL0VCO<br>0000001: CK_PLL0P = CK_PLL0VCO / 2<br>0000010: CK_PLL0P = CK_PLL0VCO / 3<br>0000011: CK_PLL0P = CK_PLL0VCO / 4<br>0000100: CK_PLL0P = CK_PLL0VCO / 5<br>…<br>1111111: CK_PLL0P = CK_PLL0VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:6 | PLL0N[8:0] | The PLL0 VCO clock multiplication factor<br>Set and reset by software (only use word / half-word write) when the PLL0 is disable. These bits used to generator PLL0 VCO clock (CK_PLL0VCO) from PLL0 VCO source clock (CK_PLL0VCOSRC). The CK_PLL0VCOSRC is described in PLL0PSC bits in RCU_PLL0 register.<br>**Note:** The frequency of CK_PLL0VCO is between 150MHz to 836MHz<br>The value of PLL0N must :<br>When PLL0 fractional latch is disabled, the value of PLL0N must satisfy: 9 ≤ PLL0N ≤ 512<br>When PLL0 fractional latch is enabled, the value of PLL0N must satisfy: 12 ≤ PLL0N ≤ 508<br>000000000: Reserved<br>…<br>000000111: Reserved<br>000001000: PLL0N = 9<br>…<br>001000000: PLL0N = 65<br>001000001: PLL0N = 66<br>…<br>111111111: PLL0N = 512 |
| 5:0 | PLL0PSC[5:0] | The PLL0 VCO source clock prescaler<br>Set and reset by software when the PLL0 is disable. These bits used to generate the clock of PLL0 VCO source clock (CK_PLL0VCOSRC) from PLL0 source clock (CK_PLL0SRC) which described in PLLSEL in RCU_PLLALL register. The VCO source clock is between 1M to 16MHz<br>000000: reserved<br>000001: CK_PLL0SRC<br>000010: CK_PLL0SRC / 2<br>000011: CK_PLL0SRC / 3<br>… |

111111: CK_PLL0SRC / 63

### 6.4.3. Clock configuration register 0 (RCU_CFG0)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2C0SEL[1:0] | | APB3PSC[2:0] | | | APB4PSC[2:0] | | | Reserved | | RTCDIV[5:0] | | | | | |
| rw | | rw | | | rw | | | | | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| APB2PSC[2:0] | | | APB1PSC[2:0] | | | Reserved | | AHBPSC[3:0] | | | | SCSS[1:0] | | SCS[1:0] | |
| rw | | | rw | | | | | rw | | | | r | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | I2C0SEL[1:0] | I2C0 clock source selection<br>Set and reset by software to control the I2C0 clock source.<br>00: CK_APB1 selected as I2C0 source clock<br>01: CK_PLL2R selected as I2C0 source clock<br>10: CK_IRC64MDIV selected as I2C0 source clock<br>11: CK_LPIRC4M selected as I2C0 source clock |
| 29:27 | APB3PSC[2:0] | APB3 prescaler selection<br>Set and reset by software to control the APB3 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 26:24 | APB4PSC[2:0] | APB4 prescaler selection<br>Set and reset by software to control the APB4 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 23:22 | Reserved | Must be kept at reset value. |
| 21:16 | RTCDIV[5:0] | RTC clock divider factor<br>Set and reset by software. These bits is used to generator clock for RTC (no more than 1MHz) from HXTAL clock.<br>000000: no clock for RTC<br>000001: no clock for RTC |

| | | |
|---|---|---|
| | | 000010: CK_HXTAL / 2<br>000011: CK_HXTAL / 3<br>…<br>111111: CK_HXTAL / 63 |
| 15:13 | APB2PSC[2:0] | APB2 prescaler selection<br>Set and reset by software to control the APB2 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 12:10 | APB1PSC[2:0] | APB1 prescaler selection<br>Set and reset by software to control the APB1 clock division ratio.<br>0xx: CK_AHB selected<br>100: (CK_AHB / 2) selected<br>101: (CK_AHB / 4) selected<br>110: (CK_AHB / 8) selected<br>111: (CK_AHB / 16) selected |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:4 | AHBPSC[3:0] | AHB and AXI prescaler selection<br>Set and reset by software to control the AHB and AXI clock division ratio<br>0xxx: CK_SYS selected<br>1000: (CK_SYS / 2) selected<br>1001: (CK_SYS / 4) selected<br>1010: (CK_SYS / 8) selected<br>1011: (CK_SYS / 16) selected<br>1100: (CK_SYS / 64) selected<br>1101: (CK_SYS / 128) selected<br>1110: (CK_SYS / 256) selected<br>1111: (CK_SYS / 512) selected |
| 3:2 | SCSS[1:0] | System clock switch status<br>Set and reset by hardware to indicate the clock source of system clock.<br>00: Select CK_IRC64MDIV as the CK_SYS source<br>01: Select CK_HXTAL as the CK_SYS source<br>10: Select CK_LPIRC4M as the CK_SYS source<br>11: Select CK_PLL0P as the CK_SYS source |
| 1:0 | SCS[1:0] | System clock switch<br>Set by software to select the CK_SYS source. Because the change of CK_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC64MDIV when leaving Deep-sleep and Standby mode or HXTAL failure is detected by HXTAL clock monitor when |

HXTAL is selected directly or indirectly as the clock source of CK_SYS.

00: Select CK_IRC64MDIV as the CK_SYS source

01: Select CK_HXTAL as the CK_SYS source

10: Select CK_LPIRC4M as the CK_SYS source

11: Select CK_PLL0P as the CK_SYS source

### 6.4.4. Clock interrupt register (RCU_INT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | LCKMIC | LCKMIF | LPIRC4M STBIC | LPIRC4M STBIE | LPIRC4M STBIF | CKMIC | PLL2 STBIC | PLL1 STBIC | PLL0 STBIC | HXTAL STBIC | IRC64MS TBIC | LXTAL STBIC | IRC32K STBIC |
| | | | w | r | w | rw | r | w | w | w | w | w | w | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL2 STBIE | PLL1 STBIE | PLL0 STBIE | HXTAL STBIE | IRC64MS TBIE | LXTAL STBIE | IRC32K STBIE | CKMIF | PLL2 STBIF | PLL1 STBIF | PLL0 STBIF | HXTAL STBIF | IRC64MS TBIF | LXTAL STBIF | IRC32K STBIF |
| | rw | rw | rw | rw | rw | rw | rw | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | LCKMIC | LXTAL clock stuck interrupt clear<br>Write 1 by software to reset the LCKMIF flag.<br>0: Not reset LCKMIF flag<br>1: Reset LCKMIF flag |
| 27 | LCKMIF | LXTAL clock stuck interrupt flag<br>Set by hardware when the LXTAL clock is stuck.<br>Reset when setting the LCKMIC bit by software.<br>0: Clock operating normally<br>1: LXTAL clock stuck |
| 26 | LPIRC4MSTBIC | LPIRC4M stabilization interrupt clear<br>Write 1 by software to reset the LPIRC4MSTBIF flag.<br>0: Not reset LPIRC4MSTBIF flag<br>1: Reset LPIRC4MSTBIF flag |
| 25 | LPIRC4MSTBIE | LPIRC4M stabilization interrupt enable<br>Set and reset by software to enable/disable the LPIRC4M stabilization interrupt.<br>0: Disable the LPIRC4M stabilization interrupt<br>1: Enable the LPIRC4M stabilization interrupt |
| 24 | LPIRC4MSTBIF | LPIRC4M Clock Stuck Interrupt Flag |

| | | |
|---|---|---|
| | | Set by hardware when the LPIRC4M clock is stuck.<br>Reset when setting the LPIRC4MSTBIC bit by software.<br>0: Clock operating normally<br>1: LPIRC4M clock stuck |
| 23 | CKMIC | HXTAL clock stuck interrupt clear<br>Write 1 by software to reset the CKMIF flag.<br>0: Not reset CKMIF flag<br>1: Reset CKMIF flag |
| 22 | PLL2STBIC | PLL2 stabilization interrupt clear<br>Write 1 by software to reset the PLL2STBIF flag.<br>0: Not reset PLL2STBIF flag<br>1: Reset PLL2STBIF flag |
| 21 | PLL1STBIC | PLL1 stabilization interrupt clear<br>Write 1 by software to reset the PLL1STBIF flag.<br>0: Not reset PLL1STBIF flag<br>1: Reset PLL1STBIF flag |
| 20 | PLL0STBIC | PLL0 stabilization interrupt clear<br>Write 1 by software to reset the PLL0STBIF flag.<br>0: Not reset PLL0STBIF flag<br>1: Reset PLL0STBIF flag |
| 19 | HXTALSTBIC | HXTAL stabilization interrupt clear<br>Write 1 by software to reset the HXTALSTBIF flag.<br>0: Not reset HXTALSTBIF flag<br>1: Reset HXTALSTBIF flag |
| 18 | IRC64MSTBIC | IRC64M stabilization interrupt clear<br>Write 1 by software to reset the IRC64MSTBIF flag.<br>0: Not reset IRC64MSTBIF flag<br>1: Reset IRC64MSTBIF flag |
| 17 | LXTALSTBIC | LXTAL stabilization interrupt clear<br>Write 1 by software to reset the LXTALSTBIF flag.<br>0: Not reset LXTALSTBIF flag<br>1: Reset LXTALSTBIF flag |
| 16 | IRC32KSTBIC | IRC32K stabilization interrupt clear<br>Write 1 by software to reset the IRC32KSTBIF flag.<br>0: Not reset IRC32KSTBIF flag<br>1: Reset IRC32KSTBIF flag |
| 15 | Reserved | Must be kept at reset value. |
| 14 | PLL2STBIE | PLL2 stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL2 stabilization interrupt. |

| | | |
|---|---|---|
| | | 0: Disable the PLL2 stabilization interrupt<br>1: Enable the PLL2 stabilization interrupt |
| 13 | PLL1STBIE | PLL1 stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL1 stabilization interrupt.<br>0: Disable the PLL1 stabilization interrupt<br>1: Enable the PLL1 stabilization interrupt |
| 12 | PLL0STBIE | PLL0 stabilization interrupt enable<br>Set and reset by software to enable/disable the PLL0 stabilization interrupt.<br>0: Disable the PLL0 stabilization interrupt<br>1: Enable the PLL0 stabilization interrupt |
| 11 | HXTALSTBIE | HXTAL stabilization interrupt enable<br>Set and reset by software to enable/disable the HXTAL stabilization interrupt<br>0: Disable the HXTAL stabilization interrupt<br>1: Enable the HXTAL stabilization interrupt |
| 10 | IRC64MSTBIE | IRC64M stabilization interrupt enable<br>Set and reset by software to enable/disable the IRC64M stabilization interrupt<br>0: Disable the IRC64M stabilization interrupt<br>1: Enable the IRC64M stabilization interrupt |
| 9 | LXTALSTBIE | LXTAL stabilization interrupt enable<br>LXTAL stabilization interrupt enable/disable control<br>0: Disable the LXTAL stabilization interrupt<br>1: Enable the LXTAL stabilization interrupt |
| 8 | IRC32KSTBIE | IRC32K stabilization interrupt enable<br>IRC32K stabilization interrupt enable/disable control<br>0: Disable the IRC32K stabilization interrupt<br>1: Enable the IRC32K stabilization interrupt |
| 7 | CKMIF | HXTAL clock stuck interrupt flag<br>Set by hardware when the HXTAL clock is stuck.<br>Reset when setting the CKMIC bit by software.<br>0: Clock operating normally<br>1: HXTAL clock stuck |
| 6 | PLL2STBIF | PLL2 stabilization interrupt flag<br>Set by hardware when the PLL2 is stable and the PLL2STBIE bit is set.<br>Reset when setting the PLL2STBIC bit by software.<br>0: No PLL2 stabilization interrupt generated<br>1: PLL2 stabilization interrupt generated |
| 5 | PLL1STBIF | PLL1 stabilization interrupt flag<br>Set by hardware when the PLL1 is stable and the PLL1STBIE bit is set.<br>Reset when setting the PLL1STBIC bit by software.<br>0: No PLL1 stabilization interrupt generated |

| | | |
|---|---|---|
| | | 1: PLL1 stabilization interrupt generated |
| 4 | PLL0STBIF | PLL0 stabilization interrupt flag<br>Set by hardware when the PLL0 is stable and the PLL0STBIE bit is set.<br>Reset when setting the PLL0STBIC bit by software.<br>0: No PLL0 stabilization interrupt generated<br>1: PLL0 stabilization interrupt generated |
| 3 | HXTALSTBIF | HXTAL stabilization interrupt flag<br>Set by hardware when the High speed 4 ~ 50 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.<br>Reset when setting the HXTALSTBIC bit by software.<br>0: No HXTAL stabilization interrupt generated<br>1: HXTAL stabilization interrupt generated |
| 2 | IRC64MSTBIF | IRC64M stabilization interrupt flag<br>Set by hardware when the Internal 64 MHz RC oscillator clock is stable and the IRC64MSTBIE bit is set.<br>Reset when setting the IRC64MSTBIC bit by software.<br>0: No IRC64M stabilization interrupt generated<br>1: IRC64M stabilization interrupt generated |
| 1 | LXTALSTBIF | LXTAL stabilization interrupt flag<br>Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set.<br>Reset when setting the LXTALSTBIC bit by software.<br>0: No LXTAL stabilization interrupt generated<br>1: LXTAL stabilization interrupt generated |
| 0 | IRC32KSTBIF | IRC32K stabilization interrupt flag<br>Set by hardware when the Internal 32kHz RC oscillator clock is stable and the IRC32KSTBIE bit is set.<br>Reset when setting the IRC32KSTBIC bit by software.<br>0: No IRC32K stabilization clock ready interrupt generated<br>1: IRC32K stabilization interrupt generated |

### 6.4.5. AHB1 reset register (RCU_AHB1RST)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | USBHS1RST | Reserved | | | | | DMAMUXRST | DMA1RST | DMA0RST | Reserved | | | | |
| | | rw | | | | | | rw | rw | rw | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Reserved | USBHS0RST | Reserved |
|---|---|---|
| | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | USBHS1RST | USBHS1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USBHS1 |
| 28:25 | Reserved | Must be kept at reset value. |
| 24 | Reserved | Must be kept at reset value. |
| 23 | DMAMUXRST | DMAMUX reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DMAMUX |
| 22 | DMA1RST | DMA1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DMA1 |
| 21 | DMA0RST | DMA0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DMA0 |
| 20:15 | Reserved | Must be kept at reset value. |
| 14 | USBHS0RST | USBHS0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USBHS0 |
| 13:0 | Reserved | Must be kept at reset value. |

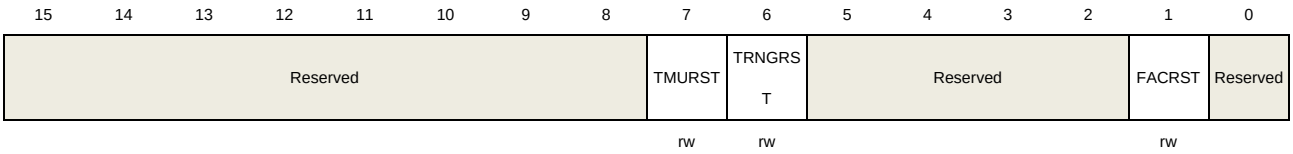
### 6.4.6. AHB2 reset register (RCU_AHB2RST)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |



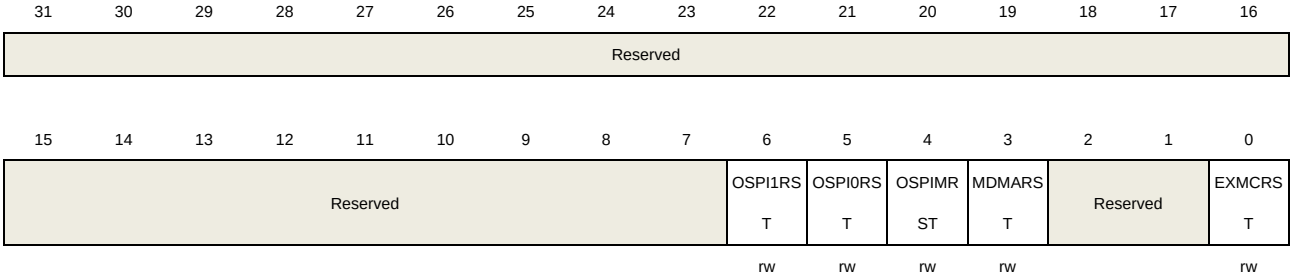

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | TMURST | TMU reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TMU |
| 6 | TRNGRST | TRNG reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TRNG |
| 5:2 | Reserved | Must be kept at reset value. |
| 1 | FACRST | FAC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the FAC |
| 1 | Reserved | Must be kept at reset value. |

### 6.4.7. AHB3 reset register (RCU_AHB3RST)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | OSPI1RST | OSPI1 reset<br>This bit is set and reset by software.<br>0: No reset |

| | | |
|---|---|---|
| | | 1: Reset the OSPI1 |
| 5 | OSPI0RST | OSPI0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the OSPI0 |
| 4 | OSPIMRST | OSPIM reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the OSPIM |
| 3 | MDMARST | MDMA reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the MDMA |
| 2:1 | Reserved | Must be kept at reset value. |
| 0 | EXMCRST | EXMC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the EXMC |

### 6.4.8. AHB4 reset register (RCU_AHB4RST)

Address offset: 0x1C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | CRCRST | Reserved | | | | | | PHRST | PGRST | PFRST | PERST | PDRST | PCRST | PBRST | PARST |
| | rw | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | CRCRST | CRC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the CRC |
| 13:8 | Reserved | Must be kept at reset value. |
| 7 | PHRST | GPIO port H reset |

This bit is set and reset by software.
0: No reset
1: Reset the GPIO port H

| | | |
|---|---|---|
| 6 | PGRST | GPIO port G reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port G |
| 5 | PFRST | GPIO port F reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port F |
| 4 | PERST | GPIO port E reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port E |
| 3 | PDRST | GPIO port D reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port D |
| 2 | PCRST | GPIO port C reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port C |
| 1 | PBRST | GPIO port B reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port B |
| 0 | PARST | GPIO port A reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the GPIO port A |

### 6.4.9. APB1 reset register (RCU_APB1RST)

Address offset: 0x20
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UART7RST | UART6RST | DACRST | DACHOLDRST | CTCRST | Reserved | | I2C3RST | I2C2RST | I2C1RST | I2C0RST | UART4RST | UART3RST | USART2RST | USART1RST | Reserved |

| rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SPI2RST | SPI1RST | Reserved | | TIMER51RST | TIMER50RST | Reserved | | TIMER23RST | TIMER22RST | TIMER6RST | TIMER5RST | TIMER4RST | TIMER3RST | TIMER2RST | TIMER1RST |
| rw | rw | | | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UART7RST | UART7 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART7 |
| 30 | UART6RST | UART6 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART6 |
| 29 | DACRST | DAC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the DAC |
| 28 | DACHOLDRST | DAC hold clock reset<br>This bit is set and reset by software. The hold clock source is IRC32K.<br>0: No reset<br>1: Reset the hold clock |
| 27 | CTCRST | CTC reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the CTC |
| 26:25 | Reserved | Must be kept at reset value. |
| 24 | I2C3RST | I2C3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C3 |
| 23 | I2C2RST | I2C2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C2 |
| 22 | I2C1RST | I2C1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C1 |

| | | |
|---|---|---|
| 21 | I2C0RST | I2C0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the I2C0 |
| 20 | UART4RST | UART4 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART4 |
| 19 | UART3RST | UART3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the UART3 |
| 18 | USART2RST | USART2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART2 |
| 17 | USART1RST | USART1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART1 |
| 16 | Reserved | Must be kept at reset value. |
| 15 | SPI2RST | SPI2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI2 |
| 14 | SPI1RST | SPI1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI1 |
| 13:12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51RST | TIMER51 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER51 |
| 10 | TIMER50RST | TIMER50 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER50 |

| | | |
|---|---|---|
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | TIMER23RST | TIMER23 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER23 |
| 6 | TIMER22RST | TIMER22 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER22 |
| 5 | TIMER6RST | TIMER6 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER6 |
| 4 | TIMER5RST | TIMER5 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER5 |
| 3 | TIMER4RST | TIMER4 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER4 |
| 2 | TIMER3RST | TIMER3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER3 |
| 1 | TIMER2RST | TIMER2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER2 |
| 0 | TIMER1RST | TIMER1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER1 |

### 6.4.10. APB2 reset register (RCU_APB2RST)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSELRST | EDOUTRST | TIMER44RST | TIMER43RST | TIMER42RST | TIMER41RST | TIMER40RST | Reserved | | | SPI5RST | SPI4RST | HPDFRST | TIMER16RST | TIMER15RST | TIMER14RST |
| rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SPI3RST | SPI0RST | Reserved | ADC2RST | ADC1RST | ADC0RST | Reserved | | USART5RST | USART0RST | Reserved | | TIMER7RST | TIMER0RST |
| | | rw | rw | | rw | rw | rw | | | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TRIGSELRST | TRIGSEL reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TRIGSEL |
| 30 | EDOUTRST | EDOUT reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the EDOUT |
| 29 | TIMER44RST | TIMER44 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER44 |
| 28 | TIMER43RST | TIMER43 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER43 |
| 27 | TIMER42RST | TIMER42 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER42 |
| 26 | TIMER41RST | TIMER41 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER41 |
| 25 | TIMER40RST | TIMER40 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER40 |
| 24:22 | Reserved | Must be kept at reset value. |
| 21 | SPI5RST | SPI5 reset<br>This bit is set and reset by software.<br>0: No reset |

| | | |
|---|---|---|
| | | 1: Reset the SPI5 |
| 20 | SPI4RST | SPI4 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI4 |
| 19 | HPDFRST | HPDF reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the HPDF |
| 18 | TIMER16RST | TIMER16 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER16 |
| 17 | TIMER15RST | TIMER15 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER15R |
| 16 | TIMER14RST | TIMER14 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER14 |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SPI3RST | SPI3 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI3 |
| 12 | SPI0RST | SPI0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SPI0 |
| 11 | Reserved | Must be kept at reset value. |
| 10 | ADC2RST | ADC2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the all ADC2 |
| 9 | ADC1RST | ADC1 reset<br>This bit is set and reset by software.<br>0: No reset |

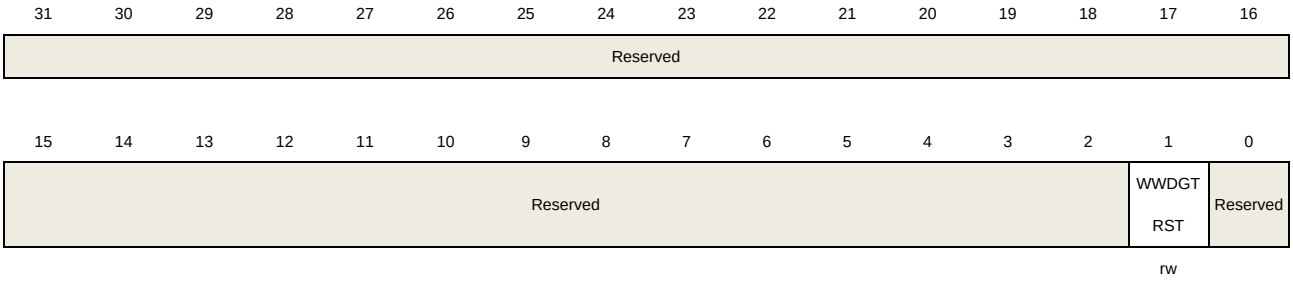
| | | |
|---|---|---|
| | | 1: Reset the all ADC1 |
| 8 | ADC0RST | ADC0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the all ADC0 |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | USART5RST | USART5 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART5 |
| 4 | USART0RST | USART0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the USART0 |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | TIMER7RST | TIMER7 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER7 |
| 0 | TIMER0RST | TIMER0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the TIMER0 |

### 6.4.11. APB3 reset register (RCU_APB3RST)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 1 | WWDGTRST | WWDGT reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the WWDGT |
| 0 | Reserved | Must be kept at reset value. |

### 6.4.12. APB4 reset register (RCU_APB4RST)

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | PMURST | LPDTSRST | VREFRST | CMPRST | SYSCFGRST |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | PMURST | PMU reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the PMU |
| 3 | LPDTSRST | LPDTS reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the LPDTS |
| 2 | VREFRST | VREF reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the VREF |
| 1 | CMPRST | CMP reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the CMP |
| 0 | SYSCFGRST | SYSCFG reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset the SYSCFG |

### 6.4.13. AHB1 enable register (RCU_AHB1EN)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USBHS1ULPIEN | USBHS1EN | Reserved | | | | | DMAMUXEN | DMA1EN | DMA0EN | Reserved | | | | |
| | rw | rw | | | | | | rw | rw | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USBHS0ULPIEN | USBHS0EN | Reserved | | | | | | | | | | | | | |
| rw | rw | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | USBHS1ULPIEN | USBHS1 ULPI clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS1 ULPI clock<br>1: Enabled USBHS1 ULPI clock |
| 29 | USBHS1EN | USBHS1 clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS1 clock<br>1: Enabled USBHS1 clock |
| 28:24 | Reserved | Must be kept at reset value. |
| 23 | DMAMUXEN | DMAMUX clock enable<br>This bit is set and reset by software.<br>0: Disabled DMAMUX clock<br>1: Enabled DMAMUX clock |
| 22 | DMA1EN | DMA1 clock enable<br>This bit is set and reset by software.<br>0: Disabled DMA1 clock<br>1: Enabled DMA1 clock |
| 21 | DMA0EN | DMA0 clock enable<br>This bit is set and reset by software.<br>0: Disabled DMA0 clock<br>1: Enabled DMA0 clock |
| 20:16 | Reserved | Must be kept at reset value. |
| 15 | USBHS0ULPIEN | USBHS0 ULPI clock enable<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled USBHS0 ULPI clock<br>1: Enabled USBHS0 ULPI clock |
| 14 | USBHS0EN | USBHS0 clock enable<br>This bit is set and reset by software.<br>0: Disabled USBHS0 clock<br>1: Enabled USBHS0 clock |
| 13:0 | Reserved | Must be kept at reset value. |

### 6.4.14. AHB2 enable register (RCU_AHB2EN)

Address offset: 0x34

Reset value: 0x0000 0100

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | RAMECCMU1EN | TMUEN | TRNGEN | Reserved | | | | FACEN | Reserved |
| | | | | | | | rw | rw | rw | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | RAMECCMU1EN | RAMECCMU1 clock enable<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU1 clock<br>1: Enabled RAMECCMU1 clock |
| 7 | TMUEN | TMU clock enable<br>This bit is set and reset by software.<br>0: Disabled TMU clock<br>1: Enabled TMU clock |
| 6 | TRNGEN | TRNG clock enable<br>This bit is set and reset by software.<br>0: Disabled TRNG clock<br>1: Enabled TRNG clock |
| 5:2 | Reserved | Must be kept at reset value. |
| 1 | FACEN | FAC clock enable<br>This bit is set and reset by software.<br>0: Disabled FAC clock |

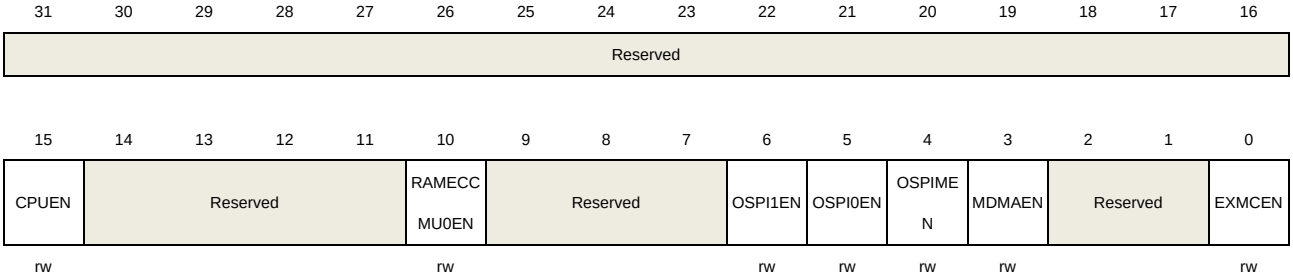
| | | |
|---|---|---|
| | | 1: Enabled FAC clock |
| 0 | Reserved | Must be kept at reset value. |

### 6.4.15. AHB3 enable register (RCU_AHB3EN)

Address offset: 0x38

Reset value: 0x0000 8400

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | CPUEN | CPU clock enable<br>This bit is set and reset by software.<br>0: Disabled CPU clock<br>1: Enabled CPU clock |
| 14:11 | Reserved | Must be kept at reset value. |
| 10 | RAMECCMU0EN | RAMECCMU0 clock enable<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU0 clock<br>1: Enabled RAMECCMU0 clock |
| 9:7 | Reserved | Must be kept at reset value. |
| 6 | OSPI1EN | OSPI1 clock enable<br>This bit is set and reset by software.<br>0: Disabled OSPI1 clock<br>1: Enabled OSPI1 clock |
| 5 | OSPI0EN | OSPI0 clock enable<br>This bit is set and reset by software.<br>0: Disabled OSPI0 clock<br>1: Enabled OSPI0 clock |
| 4 | OSPIMEN | OSPIM clock enable<br>This bit is set and reset by software.<br>0: Disabled OSPIM clock |

| | | |
|---|---|---|
| | | 1: Enabled OSPIM clock |
| 3 | MDMAEN | MDMA clock enable<br>This bit is set and reset by software.<br>0: Disabled MDMA clock<br>1: Enabled MDMA clock |
| 2:1 | Reserved | Must be kept at reset value. |
| 0 | EXMCEN | EXMC clock enable<br>This bit is set and reset by software.<br>0: Disabled EXMC clock<br>1: Enabled EXMC clock |

### 6.4.16. AHB4 enable register (RCU_AHB4EN)

Address offset: 0x3C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | CRCEN | BKPSRAMEN | Reserved | | | | | PHEN | PGEN | PFEN | PEEN | PDEN | PCEN | PBEN | PAEN |
| | rw | rw | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | CRCEN | CRC clock enable<br>This bit is set and reset by software.<br>0: Disabled CRC clock<br>1: Enabled CRC clock |
| 13 | BKPSRAMEN | Backup SRAM clock enable<br>This bit is set and reset by software.<br>0: Disabled Backup SRAM clock<br>1: Enabled Backup SRAM clock |
| 12:8 | Reserved | Must be kept at reset value. |
| 7 | PHEN | GPIO port H clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port H clock<br>1: Enabled GPIO port H clock |

| | | |
|---|---|---|
| 6 | PGEN | GPIO port G clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port G clock<br>1: Enabled GPIO port G clock |
| 5 | PFEN | GPIO port F clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port F clock<br>1: Enabled GPIO port F clock |
| 4 | PEEN | GPIO port E clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port E clock<br>1: Enabled GPIO port E clock |
| 3 | PDEN | GPIO port D clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port D clock<br>1: Enabled GPIO port D clock |
| 2 | PCEN | GPIO port C clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port C clock<br>1: Enabled GPIO port C clock |
| 1 | PBEN | GPIO port B clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port B clock<br>1: Enabled GPIO port B clock |
| 0 | PAEN | GPIO port A clock enable<br>This bit is set and reset by software.<br>0: Disabled GPIO port A clock<br>1: Enabled GPIO port A clock |

### 6.4.17. APB1 enable register (RCU_APB1EN)

Address offset: 0x40

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UART7EN | UART6EN | DACEN | DACHOLDEN | CTCEN | Reserved | | I2C3EN | I2C2EN | I2C1EN | I2C0EN | UART4EN | UART3EN | USART2EN | USART1EN | Reserved |
| rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| SPI2EN | SPI1EN | Reserved | TIMER51 EN | TIMER50 EN | Reserved | TIMER23 EN | TIMER22 EN | TIMER6EN | TIMER5EN | TIMER4EN | TIMER3EN | TIMER2EN | TIMER1EN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rw | rw | | rw | rw | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UART7EN | UART7 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART7 clock<br>1: Enabled UART7 clock |
| 30 | UART6EN | UART6 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART6 clock<br>1: Enabled UART6 clock |
| 29 | DACEN | DAC clock enable<br>This bit is set and reset by software.<br>0: Disabled DAC clock<br>1: Enabled DAC clock |
| 28 | DACHOLDEN | DAC hold clock enable<br>This bit is set and reset by software. The hold clock source is IRC32K.<br>0: Disabled DAC hold clock<br>1: Enabled DAC hold clock |
| 27 | CTCEN | CTC clock enable<br>This bit is set and reset by software.<br>0: Disabled CTC clock<br>1: Enabled CTC clock |
| 26:25 | Reserved | Must be kept at reset value. |
| 24 | I2C3EN | I2C3 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C3 clock<br>1: Enabled I2C3 clock |
| 23 | I2C2EN | I2C2 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C2 clock<br>1: Enabled I2C2 clock |
| 22 | I2C1EN | I2C1 clock enable<br>This bit is set and reset by software.<br>0: Disabled I2C1 clock<br>1: Enabled I2C1 clock |
| 21 | I2C0EN | I2C0 clock enable<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled I2C0 clock<br>1: Enabled I2C0 clock |
| 20 | UART4EN | UART4 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART4 clock<br>1: Enabled UART4 clock |
| 19 | UART3EN | UART3 clock enable<br>This bit is set and reset by software.<br>0: Disabled UART3 clock<br>1: Enabled UART3 clock |
| 18 | USART2EN | USART2 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART2 clock<br>1: Enabled USART2 clock |
| 17 | USART1EN | USART1 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART1 clock<br>1: Enabled USART1 clock |
| 16 | Reserved | Must be kept at reset value. |
| 15 | SPI2EN | SPI2 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI2 clock<br>1: Enabled SPI2 clock |
| 14 | SPI1EN | SPI1 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI1 clock<br>1: Enabled SPI1 clock |
| 13:12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51EN | TIMER51 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER51 clock<br>1: Enabled TIMER51 clock |
| 10 | TIMER50EN | TIMER50 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER50 clock<br>1: Enabled TIMER50 clock |
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | TIMER23EN | TIMER23 clock enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled TIMER23 clock<br>1: Enabled TIMER23 clock |
| 6 | TIMER22EN | TIMER22 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER22 clock<br>1: Enabled TIMER22 clock |
| 5 | TIMER6EN | TIMER6 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER6 clock<br>1: Enabled TIMER6 clock |
| 4 | TIMER5EN | TIMER5 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER5 clock<br>1: Enabled TIMER5 clock |
| 3 | TIMER4EN | TIMER4 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER4 clock<br>1: Enabled TIMER4 clock |
| 2 | TIMER3EN | TIMER3 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER3 clock<br>1: Enabled TIMER3 clock |
| 1 | TIMER2EN | TIMER2 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER2 clock<br>1: Enabled TIMER2 clock |
| 0 | TIMER1EN | TIMER1 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER1 clock<br>1: Enabled TIMER1 clock |

### 6.4.18. APB2 enable register (RCU_APB2EN)

Address offset: 0x44

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSEL EN | EDOUTE N | TIMER44 EN | TIMER43 EN | TIMER42 EN | TIMER41 EN | TIMER40 EN | Reserved | | | SPI5EN | SPI4EN | HPDFEN | TIMER16 EN | TIMER15 EN | TIMER14 EN |

| rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw | rw |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI3EN | SPI0EN | Reserved | ADC2EN | ADC1EN | ADC0EN | Reserved | | USART5EN | USART0EN | Reserved | | TIMER7EN | TIMER0EN |
| | | rw | rw | | rw | rw | rw | | | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TRIGSELEN | TRIGSEL clock enable<br>This bit is set and reset by software.<br>0: Disabled TRIGSEL clock<br>1: Enabled TRIGSEL clock |
| 30 | EDOUTEN | EDOUT clock enable<br>This bit is set and reset by software.<br>0: Disabled EDOUT clock<br>1: Enabled EDOUT clock |
| 29 | TIMER44EN | TIMER44 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER44 clock<br>1: Enabled TIMER44 clock |
| 28 | TIMER43EN | TIMER43 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER43 clock<br>1: Enabled TIMER43 clock |
| 27 | TIMER42EN | TIMER42 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER42 clock<br>1: Enabled TIMER42 clock |
| 26 | TIMER41EN | TIMER41 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER41 clock<br>1: Enabled TIMER41 clock |
| 25 | TIMER40EN | TIMER40 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER40 clock<br>1: Enabled TIMER40 clock |
| 24:22 | Reserved | Must be kept at reset value. |
| 21 | SPI5EN | SPI5 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI5 clock<br>1: Enabled SPI5 clock |

| | | |
|---|---|---|
| 20 | SPI4EN | SPI4 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI4 clock<br>1: Enabled SPI4 clock |
| 19 | HPDFEN | HPDF clock enable<br>This bit is set and reset by software.<br>0: Disabled HPDF clock<br>1: Enabled HPDF clock |
| 18 | TIMER16EN | TIMER16 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER16 clock<br>1: Enabled TIMER16 clock |
| 17 | TIMER15EN | TIMER15 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER15 clock<br>1: Enabled TIMER15 clock |
| 16 | TIMER14EN | TIMER14 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER14 clock<br>1: Enabled TIMER14 clock |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SPI3EN | SPI3 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI3 clock<br>1: Enabled SPI3 clock |
| 12 | SPI0EN | SPI0 clock enable<br>This bit is set and reset by software.<br>0: Disabled SPI0 clock<br>1: Enabled SPI0 clock |
| 11 | Reserved | Must be kept at reset value. |
| 10 | ADC2EN | ADC2 clock enable<br>This bit is set and reset by software.<br>0: Disabled ADC2 clock<br>1: Enabled ADC2 clock |
| 9 | ADC1EN | ADC1 clock enable<br>This bit is set and reset by software.<br>0: Disabled ADC1 clock<br>1: Enabled ADC1 clock |
| 8 | ADC0EN | ADC0 clock enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled ADC0 clock<br>1: Enabled ADC0 clock |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | USART5EN | USART5 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART5 clock<br>1: Enabled USART5 clock |
| 4 | USART0EN | USART0 clock enable<br>This bit is set and reset by software.<br>0: Disabled USART0 clock<br>1: Enabled USART0 clock |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | TIMER7EN | TIMER7 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER7 clock<br>1: Enabled TIMER7 clock |
| 0 | TIMER0EN | TIMER0 clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER0 clock<br>1: Enabled TIMER0 clock |

### 6.4.19. APB3 enable register (RCU_APB3EN)

Address offset: 0x48

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | WWDGTEN | Reserved |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WWDGTEN | WWDGT clock enable<br>This bit is set and reset by software.<br>0: Disabled WWDGT clock |

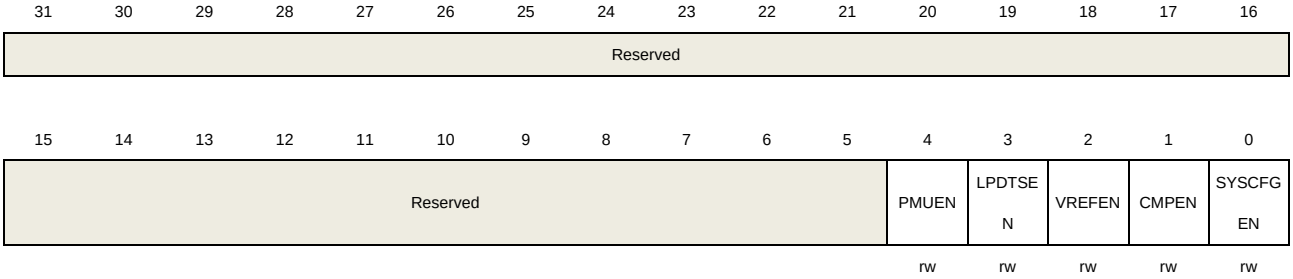
| | | |
|---|---|---|
| | | 1: Enabled WWDGT clock |
| 0 | Reserved | Must be kept at reset value. |

## 6.4.20. APB4 enable register (RCU_APB4EN)

Address offset: 0x4C

Reset value: 0x0000 0010

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



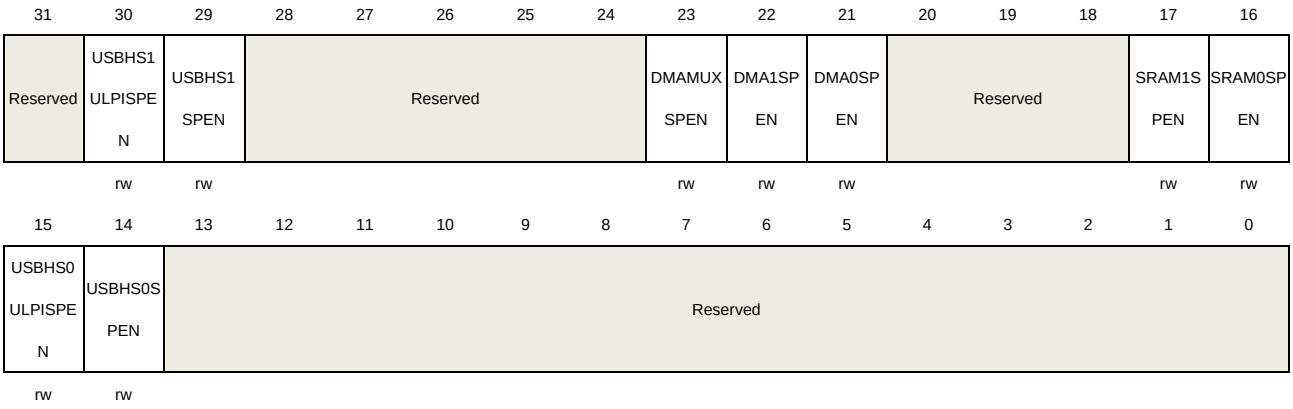

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | PMUEN | PMU clock enable<br>This bit is set and reset by software.<br>0: Disabled PMU clock<br>1: Enabled PMU clock |
| 3 | LPDTSEN | LPDTS clock enable<br>This bit is set and reset by software.<br>0: Disabled LPDTS clock<br>1: Enabled LPDTS clock |
| 2 | VREFEN | VREF clock enable<br>This bit is set and reset by software.<br>0: Disabled VREF clock<br>1: Enabled VREF clock |
| 1 | CMPEN | CMP clock enable<br>This bit is set and reset by software.<br>0: Disabled CMP clock<br>1: Enabled CMP clock |
| 0 | SYSCFGEN | SYSCFG clock enable<br>This bit is set and reset by software.<br>0: Disabled SYSCFG clock<br>1: Enabled SYSCFG clock |

## 6.4.21. AHB1 sleep mode enable register (RCU_AHB1SPEN)

Address offset: 0x50

Reset value: 0x7EE3 C00F

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | USBHS1ULPISPEN | USBHS1 ULPI clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS1 ULPI clock when sleep mode<br>1: Enabled USBHS1 ULPI clock when sleep mode |
| 29 | USBHS1SPEN | USBHS1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS1 clock when sleep mode<br>1: Enabled USBHS1 clock when sleep mode |
| 28:24 | Reserved | Must be kept at reset value. |
| 23 | DMAMUXSPEN | DMAMUX clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DMAMUX clock when sleep mode<br>1: Enabled DMAMUX clock when sleep mode |
| 22 | DMA1SPEN | DMA1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DMA1 clock when sleep mode<br>1: Enabled DMA1 clock when sleep mode |
| 21 | DMA0SPEN | DMA0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DMA0 clock when sleep mode<br>1: Enabled DMA0 clock when sleep mode |
| 20:18 | Reserved | Must be kept at reset value. |
| 17 | SRAM1SPEN | SRAM1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SRAM1 clock when sleep mode |

| | | |
|---|---|---|
| | | 1: Enabled SRAM1 clock when sleep mode |
| 16 | SRAM0SPEN | SRAM0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SRAM0 clock when sleep mode<br>1: Enabled SRAM0 clock when sleep mode |
| 15 | USBHS0ULPISPEN | USBHS0 ULPI clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS0 ULPI clock when sleep mode<br>1: Enabled USBHS0 ULPI clock when sleep mode |
| 14 | USBHS0SPEN | USBHS0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USBHS0 clock when sleep mode<br>1: Enabled USBHS0 clock when sleep mode |
| 13:0 | Reserved | Must be kept at reset value. |

## 6.4.22. AHB2 sleep mode enable register (RCU_AHB2SPEN)

Address offset: 0x54

Reset value: 0x0000 01DF

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | RAMECCMU1SPEN | TMUSPEN | TRNGSPEN | Reserved | | | | FACSPEN | Reserved |
| | | | | | | | rw | rw | rw | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | RAMECCMU1SPEN | RAMECCMU1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU1 clock when sleep mode<br>1: Enabled RAMECCMU1 clock when sleep mode |
| 7 | TMUSPEN | TMU clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TMU clock when sleep mode<br>1: Enabled TMU clock when sleep mode |
| 6 | TRNGSPEN | TRNG clock enable when sleep mode<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled TRNG clock when sleep mode<br>1: Enabled TRNG clock when sleep mode |
| 5:2 | Reserved | Must be kept at reset value. |
| 1 | FACSPEN | FAC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled FAC clock when sleep mode<br>1: Enabled FAC clock when sleep mode |
| 0 | Reserved | Must be kept at reset value. |

### 6.4.23. AHB3 sleep mode enable register (RCU_AHB3SPEN)

Address offset: 0x58

Reset value: 0x0000 C77F

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMCSPEN | AXISRAMSPEN | Reserved | | | RAMECCMU0SPEN | Reserved | | | OSPI1SPEN | OSPI0SPEN | OSPIMSPEN | MDMASPEN | Reserved | | EXMCSPEN |
| rw | rw | | | | rw | | | | rw | rw | rw | rw | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | FMCSPEN | FMC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled FMC clock when sleep mode<br>1: Enabled FMC clock when sleep mode |
| 14 | AXISRAMSPEN | AXI SRAM clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled AXI SRAM clock when sleep mode<br>1: Enabled AXI SRAM clock when sleep mode |
| 13:11 | Reserved | Must be kept at reset value. |
| 10 | RAMECCMU0SPEN | RAMECCMU0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled RAMECCMU0 clock when sleep mode<br>1: Enabled RAMECCMU0 clock when sleep mode |
| 9:7 | Reserved | Must be kept at reset value. |
| 6 | OSPI1SPEN | OSPI1 clock enable when sleep mode |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled OSPI1 clock when sleep mode<br>1: Enabled OSPI1 clock when sleep mode |
| 5 | OSPI0SPEN | OSPI0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled OSPI0 clock when sleep mode<br>1: Enabled OSPI0 clock when sleep mode |
| 4 | OSPIMSPEN | OSPIM clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled OSPIM clock when sleep mode<br>1: Enabled OSPIM clock when sleep mode |
| 3 | MDMASPEN | MDMA clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled MDMA clock when sleep mode<br>1: Enabled MDMA clock when sleep mode |
| 2:1 | Reserved | Must be kept at reset value. |
| 0 | EXMCSPEN | EXMC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled EXMC clock when sleep mode<br>1: Enabled EXMC clock when sleep mode |

### 6.4.24. AHB4 sleep mode enable register (RCU_AHB4SPEN)

Address offset: 0x5C

Reset value: 0x0000 63FF

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | CRCSPEN | BKPSRAMSPEN | Reserved | | | | | PHSPEN | PGSPEN | PFSPEN | PESPEN | PDSPEN | PCSPEN | PBSPEN | PASPEN |
| | rw | rw | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | CRCSPEN | CRC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CRC clock when sleep mode<br>1: Enabled CRC clock when sleep mode |

| | | |
|---|---|---|
| 13 | BKPSRAMSPEN | Backup SRAM clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled Backup SRAM clock when sleep mode<br>1: Enabled Backup SRAM clock when sleep mode |
| 12:8 | Reserved | Must be kept at reset value. |
| 7 | PHSPEN | GPIO port H clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port H clock when sleep mode<br>1: Enabled GPIO port H clock when sleep mode |
| 6 | PGSPEN | GPIO port G clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port G clock when sleep mode<br>1: Enabled GPIO port G clock when sleep mode |
| 5 | PFSPEN | GPIO port F clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port F clock when sleep mode<br>1: Enabled GPIO port F clock when sleep mode |
| 4 | PESPEN | GPIO port E clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port E clock when sleep mode<br>1: Enabled GPIO port E clock when sleep mode |
| 3 | PDSPEN | GPIO port D clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port D clock when sleep mode<br>1: Enabled GPIO port D clock when sleep mode |
| 2 | PCSPEN | GPIO port C clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port C clock when sleep mode<br>1: Enabled GPIO port C clock when sleep mode |
| 1 | PBSPEN | GPIO port B clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port B clock when sleep mode<br>1: Enabled GPIO port B clock when sleep mode |
| 0 | PASPEN | GPIO port A clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled GPIO port A clock when sleep mode<br>1: Enabled GPIO port A clock when sleep mode |

### 6.4.25. APB1 sleep mode enable register (RCU_APB1SPEN)

Address offset: 0x60

Reset value: 0xF9FF EFFF

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UART7SPEN | UART6SPEN | DACSPEN | DACHOLDSPEN | CTCSPEN | Reserved | | I2C3SPEN | I2C2SPEN | I2C1SPEN | I2C0SPEN | UART4SPEN | UART3SPEN | USART2SPEN | USART1SPEN | Reserved |
| rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPI2SPEN | SPI1SPEN | Reserved | | TIMER51SPEN | TIMER50SPEN | Reserved | | TIMER23SPEN | TIMER22SPEN | TIMER6SPEN | TIMER5SPEN | TIMER4SPEN | TIMER3SPEN | TIMER2SPEN | TIMER1SPEN |
| rw | rw | | | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UART7SPEN | UART7 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART7 clock when sleep mode<br>1: Enabled UART7 clock when sleep mode |
| 30 | UART6SPEN | UART6 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART6 clock when sleep mode<br>1: Enabled UART6 clock when sleep mode |
| 29 | DACSPEN | DAC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled DAC clock when sleep mode<br>1: Enabled DAC clock when sleep mode |
| 28 | DACHOLDSPEN | DAC hold clock enable when sleep mode<br>This bit is set and reset by software. The hold clock source is IRC32K.<br>0: Disabled DAC hold clock when sleep mode<br>1: Enabled DAC hold clock when sleep mode |
| 27 | CTCSPEN | CTC clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CTC clock when sleep mode<br>1: Enabled CTC clock when sleep mode |
| 26:25 | Reserved | Must be kept at reset value. |
| 24 | I2C3SPEN | I2C3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C3 clock when sleep mode<br>1: Enabled I2C3 clock when sleep mode |

| | | |
|---|---|---|
| 23 | I2C2SPEN | I2C2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C2 clock when sleep mode<br>1: Enabled I2C2 clock when sleep mode |
| 22 | I2C1SPEN | I2C1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C1 clock when sleep mode<br>1: Enabled I2C1 clock when sleep mode |
| 21 | I2C0SPEN | I2C0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled I2C0 clock when sleep mode<br>1: Enabled I2C0 clock when sleep mode |
| 20 | UART4SPEN | UART4 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART4 clock when sleep mode<br>1: Enabled UART4 clock when sleep mode |
| 19 | UART3SPEN | UART3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled UART3 clock when sleep mode<br>1: Enabled UART3 clock when sleep mode |
| 18 | USART2SPEN | USART2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART2 clock when sleep mode<br>1: Enabled USART2 clock when sleep mode |
| 17 | USART1SPEN | USART1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART1 clock when sleep mode<br>1: Enabled USART1 clock when sleep mode |
| 16 | Reserved | Must be kept at reset value. |
| 15 | SPI2SPEN | SPI2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI2 clock when sleep mode<br>1: Enabled SPI2 clock when sleep mode |
| 14 | SPI1SPEN | SPI1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI1 clock when sleep mode<br>1: Enabled SPI1 clock when sleep mode |
| 13:12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51SPEN | TIMER51 clock enable when sleep mode |

| | | |
|---|---|---|
| | | This bit is set and reset by software.<br>0: Disabled TIMER51 clock when sleep mode<br>1: Enabled TIMER51 clock when sleep mode |
| 10 | TIMER50SPEN | TIMER50 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER50 clock when sleep mode<br>1: Enabled TIMER50 clock when sleep mode |
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | TIMER23SPEN | TIMER23 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER23 clock when sleep mode<br>1: Enabled TIMER23 clock when sleep mode |
| 6 | TIMER22SPEN | TIMER22 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER22 clock when sleep mode<br>1: Enabled TIMER22 clock when sleep mode |
| 5 | TIMER6SPEN | TIMER6 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER6 clock when sleep mode<br>1: Enabled TIMER6 clock when sleep mode |
| 4 | TIMER5SPEN | TIMER5 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER5 clock when sleep mode<br>1: Enabled TIMER5 clock when sleep mode |
| 3 | TIMER4SPEN | TIMER4 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER4 clock when sleep mode<br>1: Enabled TIMER4 clock when sleep mode |
| 2 | TIMER3SPEN | TIMER3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER3 clock when sleep mode<br>1: Enabled TIMER3 clock when sleep mode |
| 1 | TIMER2SPEN | TIMER2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER2 clock when sleep mode<br>1: Enabled TIMER2 clock when sleep mode |
| 0 | TIMER1SPEN | TIMER1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER1 clock when sleep mode |

1: Enabled TIMER1 clock when sleep mode

## 6.4.26. APB2 sleep mode enable register (RCU_APB2SPEN)

Address offset: 0x64

Reset value: 0xFFFF 3733

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRIGSELSPEN | EDOUTSPEN | TIMER44SPEN | TIMER43SPEN | TIMER42SPEN | TIMER41SPEN | TIMER40SPEN | Reserved | | | SPI5SPEN | SPI4SPEN | HPDFSPEN | TIMER16SPEN | TIMER15SPEN | TIMER14SPEN |
| rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | SPI3SPEN | SPI0SPEN | Reserved | ADC2SPEN | ADC1SPEN | ADC0SPEN | Reserved | | USART5SPEN | USART0SPEN | Reserved | | TIMER7SPEN | TIMER0SPEN |
| | | rw | rw | | rw | rw | rw | | | rw | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TRIGSELSPEN | TRIGSEL clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TRIGSEL clock when sleep mode<br>1: Enabled TRIGSEL clock when sleep mode |
| 30 | EDOUTSPEN | EDOUT clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled EDOUT clock when sleep mode<br>1: Enabled EDOUT clock when sleep mode |
| 29 | TIMER44SPEN | TIMER44 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER44 clock when sleep mode<br>1: Enabled TIMER44 clock when sleep mode |
| 28 | TIMER43SPEN | TIMER43 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER43 clock when sleep mode<br>1: Enabled TIMER43 clock when sleep mode |
| 27 | TIMER42SPEN | TIMER42 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER42 clock when sleep mode<br>1: Enabled TIMER42 clock when sleep mode |
| 26 | TIMER41SPEN | TIMER41 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER41 clock when sleep mode |

| | | |
|---|---|---|
| | | 1: Enabled TIMER41 clock when sleep mode |
| 25 | TIMER40SPEN | TIMER40 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER40 clock when sleep mode<br>1: Enabled TIMER40 clock when sleep mode |
| 24:22 | Reserved | Must be kept at reset value. |
| 21 | SPI5SPEN | SPI5 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI5 clock when sleep mode<br>1: Enabled SPI5 clock when sleep mode |
| 20 | SPI4SPEN | SPI4 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI4 clock when sleep mode<br>1: Enabled SPI4 clock when sleep mode |
| 19 | HPDFSPEN | HPDF clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled HPDF clock when sleep mode<br>1: Enabled HPDF clock when sleep mode |
| 18 | TIMER16SPEN | TIMER16 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER16 clock when sleep mode<br>1: Enabled TIMER16 clock when sleep mode |
| 17 | TIMER15SPEN | TIMER15 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER15 clock when sleep mode<br>1: Enabled TIMER15 clock when sleep mode |
| 16 | TIMER14SPEN | TIMER14 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER14 clock when sleep mode<br>1: Enabled TIMER14 clock when sleep mode |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SPI3SPEN | SPI3 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI3 clock when sleep mode<br>1: Enabled SPI3 clock when sleep mode |
| 12 | SPI0SPEN | SPI0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SPI0 clock when sleep mode |

| | | |
|---|---|---|
| | | 1: Enabled SPI0 clock when sleep mode |
| 11 | Reserved | Must be kept at reset value. |
| 10 | ADC2SPEN | ADC2 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled ADC2 clock when sleep mode<br>1: Enabled ADC2 clock when sleep mode |
| 9 | ADC1SPEN | ADC1 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled ADC1 clock when sleep mode<br>1: Enabled ADC1 clock when sleep mode |
| 8 | ADC0SPEN | ADC0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled ADC0 clock when sleep mode<br>1: Enabled ADC0 clock when sleep mode |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | USART5SPEN | USART5 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART5 clock when sleep mode<br>1: Enabled USART5 clock when sleep mode |
| 4 | USART0SPEN | USART0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled USART0 clock when sleep mode<br>1: Enabled USART0 clock when sleep mode |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | TIMER7SPEN | TIMER7 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER7 clock when sleep mode<br>1: Enabled TIMER7 clock when sleep mode |
| 0 | TIMER0SPEN | TIMER0 clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled TIMER0 clock when sleep mode<br>1: Enabled TIMER0 clock when sleep mode |

### 6.4.27. APB3 sleep mode enable register (RCU_APB3SPEN)

Address offset: 0x68

Reset value: 0x0000 0003

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | WWDGTSPEN | Reserved |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | WWDGTSPEN | WWDGT clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled WWDGT clock when sleep mode<br>1: Enabled WWDGT clock when sleep mode |
| 0 | Reserved | Must be kept at reset value. |

## 6.4.28. APB4 sleep mode enable register (RCU_APB4SPEN)

Address offset: 0x6C

Reset value: 0x0000 001F

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | PMUSPEN | LPDTSSPEN | VREFSPEN | CMPSPEN | SYSCFGSPEN |
| | | | | | | | | | | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | PMUSPEN | PMU clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled PMU clock when sleep mode<br>1: Enabled PMU clock when sleep mode |
| 3 | LPDTSSPEN | LPDTS clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled LPDTS clock when sleep mode<br>1: Enabled LPDTS clock when sleep mode |
| 2 | VREFSPEN | VREF clock enable when sleep mode<br>This bit is set and reset by software. |

| | | |
|---|---|---|
| | | 0: Disabled VREF clock when sleep mode<br>1: Enabled VREF clock when sleep mode |
| 1 | CMPSPEN | CMPclock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled CMPclock when sleep mode<br>1: Enabled CMP clock when sleep mode |
| 0 | SYSCFGSPEN | SYSCFG clock enable when sleep mode<br>This bit is set and reset by software.<br>0: Disabled SYSCFG clock when sleep mode<br>1: Enabled SYSCFG clock when sleep mode |

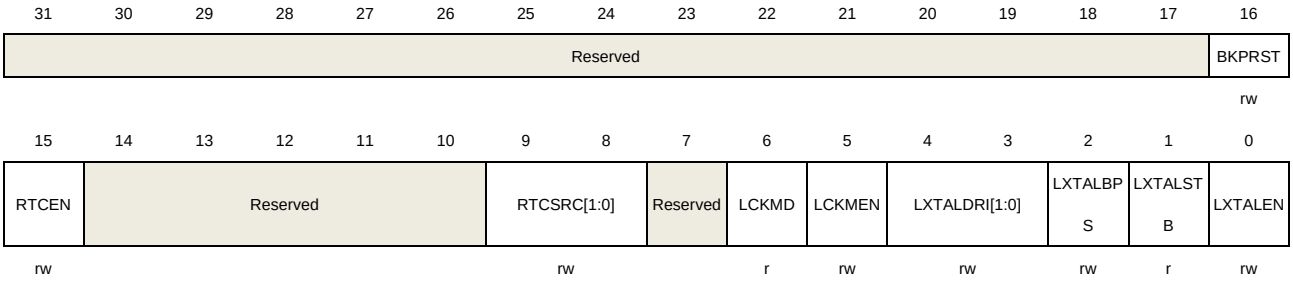
## 6.4.29. Backup domain control register (RCU_BDCTL)

Address offset: 0x70

Reset value: 0x0000 0018, reset by Backup domain Reset.

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

**Note:** The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the backup domain control register (RCU_BDCTL) are only reset after a backup domain reset. These bits can be modified only when the BKPWEN bit in the power control register (PMU_CTL) is set.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | BKPRST | Backup domain reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Resets backup domain |
| 15 | RTCEN | RTC clock enable<br>This bit is set and reset by software.<br>0: Disabled RTC clock<br>1: Enabled RTC clock |
| 14:10 | Reserved | Must be kept at reset value. |
| 9:8 | RTCSRC[1:0] | RTC clock entry selection<br>Set and reset by software to control the RTC clock source. Once the RTC clock |

| | | |
|---|---|---|
| | | source has been selected, it cannot be changed anymore unless the backup domain is reset.<br>00: No clock selected<br>01: CK_LXTAL selected as RTC source clock<br>10: CK_IRC32K selected as RTC source clock<br>11: (CK_HXTAL / RTCDIV) selected as RTC source clock, please refer to RTCDIV bits in RCU_CFG0 register. |
| 7 | Reserved | Must be kept at reset value. |
| 6 | LCKMD | LXTAL clock failure detection<br>Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator (LXTAL). It can be clean by disable LCKMEN or disable LXTALEN.<br>0: No failure detected on LXTAL (32 kHz oscillator)<br>1: Failure detected on LXTAL (32 kHz oscillator) |
| 5 | LCKMEN | LXTAL clock monitor enable<br>0: Disable the LXTAL clock monitor<br>1: Enable the LXTAL clock monitor<br>Set by software to enable the clock security system on LXTAL (32 kHz oscillator). LCKMEN should be enabled only on the LXTAL is enabled (LXTALEN bit enabled) and ready (LXTALSTB flag set by hardware).<br>**Note:** Once LCKMEN bit is set, this bit can be reset by backup domain reset or reseting this bit after detecting LXTAL clock failure (LCKMD = 1). |
| 4:3 | LXTALDRI[1:0] | LXTAL drive capability<br>Set and reset by software. Backup domain reset resets this value.<br>00: Lower driving capability<br>01: Medium low driving capability<br>10: Medium high driving capability<br>11: Higher driving capability<br>**Note:** The LXTALDRI is not in bypass mode. |
| 2 | LXTALBPS | LXTAL bypass mode enable<br>Set and reset by software.<br>0: Disable the LXTAL Bypass mode<br>1: Enable the LXTAL Bypass mode |
| 1 | LXTALSTB | Low speed crystal oscillator stabilization flag<br>Set by hardware to indicate if the LXTAL output clock is stable and ready for use.<br>0: LXTAL is not stable<br>1: LXTAL is stable |
| 0 | LXTALEN | LXTAL enable<br>Set and reset by software.<br>0: Disable LXTAL |

1: Enable LXTAL

## 6.4.30. Reset source/clock register (RCU_RSTSCK)

Address offset: 0x74

Reset value: 0x0E00 0000, all reset flags reset by power reset only, RSTFC / IRC32KEN reset by system reset.

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPRSTF | WWDGTRSTF | FWDGTRSTF | SWRSTF | PORRSTF | EPRSTF | BORRSTF | RSTFC | Reserved | | | | | | | |
| r | r | r | r | r | r | r | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | IRC32KSTB | IRC32KEN |
| | | | | | | | | | | | | | | r | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LPRSTF | Low-power reset flag<br>Set by hardware when Deep-sleep /standby reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No Low-power management reset generated<br>1: Low-power management reset generated |
| 30 | WWDGTRSTF | Window watchdog timer reset flag<br>Set by hardware when a window watchdog timer reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No window watchdog reset generated<br>1: Window watchdog reset generated |
| 29 | FWDGTRSTF | Free watchdog timer reset flag<br>Set by hardware when a free watchdog timer reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No free watchdog timer reset generated<br>1: free Watchdog timer reset generated |
| 28 | SWRSTF | Software reset flag<br>Set by hardware when a software reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No software reset generated<br>1: Software reset generated |
| 27 | PORRSTF | Power reset flag<br>Set by hardware when a power reset generated.<br>Reset by writing 1 to the RSTFC bit. |

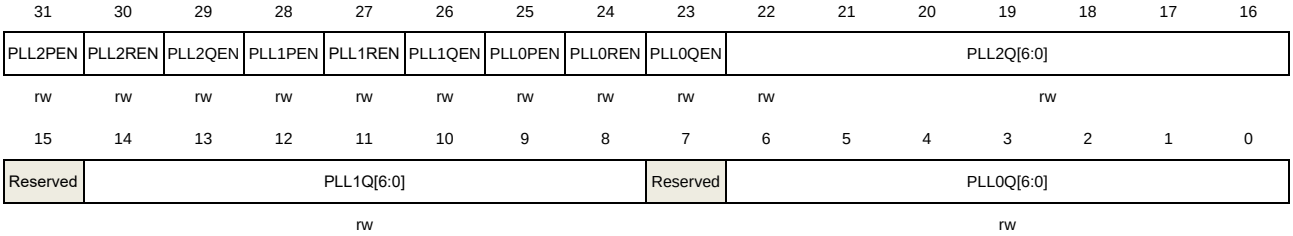
| | | |
|---|---|---|
| | | 0: No Power reset generated<br>1: Power reset generated |
| 26 | EPRSTF | External PIN reset flag<br>Set by hardware when an external pin reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No external pin reset generated<br>1: External pin reset generated |
| 25 | BORRSTF | BOR reset flag<br>Set by hardware when a BOR reset generated.<br>Reset by writing 1 to the RSTFC bit.<br>0: No BOR reset generated<br>1: BOR reset generated |
| 24 | RSTFC | Reset flag clear<br>This bit is set by software to clear all reset flags.<br>0: Not clear reset flags<br>1: Clear reset flags |
| 23:2 | Reserved | Must be kept at reset value. |
| 1 | IRC32KSTB | IRC32K stabilization flag<br>Set by hardware to indicate if the IRC32K output clock is stable and ready for use.<br>0: IRC32K is not stable<br>1: IRC32K is stable |
| 0 | IRC32KEN | IRC32K enable<br>Set and reset by software.<br>0: Disable IRC32K<br>1: Enable IRC32K |

## 6.4.31. PLL clock additional control register (RCU_PLLADDCTL)

Address offset: 0x80

Reset value: 0xFF81 0101

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLL2PEN | PLL2P divider output enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software. The PLL2PEN bit can be written only if the PLL2EN is 0.<br>0: Disable the CK_PLL2P output<br>1: Enable the CK_PLL2P output |
| 30 | PLL2REN | PLL2R divider output enable<br>This bit is set and reset by software. The PLL2REN bit can be written only if the PLL2EN is 0.<br>0: Disable the CK_PLL2R output<br>1: Enable the CK_PLL2R output |
| 29 | PLL2QEN | PLL2Q divider output enable<br>This bit is set and reset by software. The PLL2QEN bit can be written only if the PLL2EN is 0.<br>0: Disable the CK_PLL2Q output<br>1: Enable the CK_PLL2Q output |
| 28 | PLL1PEN | PLL1P divider output enable<br>This bit is set and reset by software. The PLL1PEN bit can be written only if the PLL1EN is 0.<br>0: Disable the CK_PLL1P output<br>1: Enable the CK_PLL1P output |
| 27 | PLL1REN | PLL1R divider output enable<br>This bit is set and reset by software. The PLL1REN bit can be written only if the PLL1EN is 0.<br>0: Disable the CK_PLL1R output<br>1: Enable the CK_PLL1R output |
| 26 | PLL1QEN | PLL1Q divider output enable<br>This bit is set and reset by software. The PLL1QEN bit can be written only if the PLL1EN is 0.<br>0: Disable the CK_PLL1Q output<br>1: Enable the CK_PLL1Q output |
| 25 | PLL0PEN | PLL0P divider output enable<br>This bit is set and reset by software. The PLL0PEN bit can be written only if the PLL0EN is 0.<br>0: Disable the CK_PLL0P output<br>1: Enable the CK_PLL0P output |
| 24 | PLL0REN | PLL0R divider output enable<br>This bit is set and reset by software. The PLL0REN bit can be written only if the PLL0EN is 0.<br>0: Disable the CK_PLL0R output<br>1: Enable the CK_PLL0R output |
| 23 | PLL0QEN | PLL0Q divider output enable |

| | | |
|---|---|---|
| | | This bit is set and reset by software. The PLL0QEN bit can be written only if the PLL0EN is 0.<br>0: Disable the CK_PLL0Q output<br>1: Enable the CK_PLL0Q output |
| 22:16 | PLL2Q[6:0] | The PLL2Q output frequency division factor from PLL2 VCO clock<br>Set and reset by software when the PLL2 is disable. These bits used to generator PLL2Q output clock (CK_PLL2Q) from PLL2 VCO clock (CK_PLL2VCO). The CK_PLL2VCO is described in PLL2N bits in RCU_PLL2 register.<br>0000000: CK_PLL2Q = CK_PLL2VCO<br>0000001: CK_PLL2Q = CK_PLL2VCO / 2<br>0000010: CK_PLL2Q = CK_ PLL2VCO / 3.<br>0000011: CK_PLL2Q = CK_ PLL2VCO / 4<br>0000100: CK_PLL2Q = CK_ PLL2VCO / 5<br>…<br>1111111: CK_PLL2Q = CK_ PLL2VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:8 | PLL1Q[6:0] | The PLL1Q output frequency division factor from PLL1 VCO clock<br>Set and reset by software when the PLL1 is disable. These bits used to generator PLL1Q output clock (CK_PLL1Q) from PLL1 VCO clock (CK_PLL1VCO). The CK_PLL1VCO is described in PLL1N bits in RCU_PLL1 register.<br>0000000: CK_PLL1Q = CK_PLL1VCO<br>0000001: CK_ PLL1Q = CK_PLL1VCO / 2<br>0000010: CK_PLL1Q = CK_PLL1VCO / 3.<br>0000011: CK_PLL1Q = CK_PLL1VCO / 4<br>0000100: CK_PLL1Q = CK_PLL1VCO / 5<br>…<br>1111111: CK_PLL1Q = CK_PLL1VCO / 128 |
| 7 | Reserved | Must be kept at reset value. |
| 6:0 | PLL0Q[6:0] | The PLL0Q output frequency division factor from PLL0 VCO clock<br>Set and reset by software when the PLL0 is disable. These bits used to generator PLL0Q output clock (CK_PLL0Q) from PLL0 VCO clock (CK_PLL0VCO). The CK_PLL0Q is used to USBHS (48MHz), TRNG(48MHz). The CK_PLL0VCO is described in PLL0N bits in RCU_PLL0 register.<br>0000000: CK_PLL0Q = CK_PLL0VCO<br>0000001: CK_PLL0Q = CK_PLL0VCO / 2<br>0000010: CK_PLL0Q = CK_PLL0VCO / 3<br>0000011: CK_PLL0Q = CK_PLL0VCO / 4<br>0000100: CK_PLL0Q = CK_PLL0VCO / 5<br>…<br>1111111: CK_PLL0Q = CK_PLL0VCO / 128 |

## 6.4.32. PLL1 register (RCU_PLL1)

Address offset: 0x84

Reset value: 0x0101 2020

To configure the PLL1 clock, refer to the following formula:

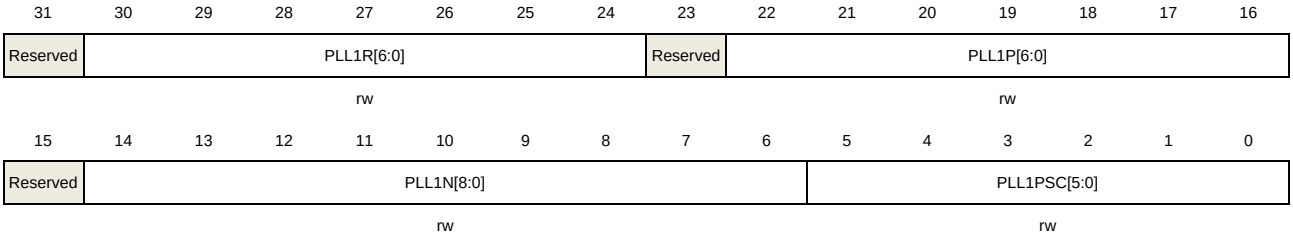
CK_PLL1VCOSRC = CK_PLL1SRC / PLL1PSC

CK_PLL1VCO = CK_PLL1VCOSRC × (PLL1N + PLL1FRAN / $2^{13}$)

CK_PLL1P = CK_ PLL1VCO / PLL1P

CK_PLL1R = CK_PLL1VCO / PLL1R

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | PLL1R[6:0] | The PLL1R output frequency division factor from PLL1 VCO clock<br>Set and reset by software when the PLL1 is disable. These bits used to generate PLL1R output clock (CK_PLL1R) from PLL1 VCO clock (CK_PLL1VCO). The CK_PLL1VCO is described in PLL1N bits in RCU_PLL1 register.<br>0000000: CK_PLL1R = CK_PLL1VCO<br>0000001: CK_PLL1R = CK_PLL1VCO / 2<br>0000010: CK_PLL1R = CK_PLL1VCO / 3.<br>0000011: CK_PLL1R = CK_PLL1VCO / 4<br>0000100: CK_PLL1R = CK_PLL1VCO / 5<br>…<br>1111111: CK_PLL1R = CK_PLL1VCO / 128 |
| 23 | Reserved | Must be kept at reset value. |
| 22:16 | PLL1P[6:0] | The PLL1P output frequency division factor from PLL1 VCO clock<br>Set and reset by software when the PLL1 is disable. These bits used to generator PLL1P output clock (CK_PLL1P) from PLL1 VCO clock (CK_PLL1VCO). The CK_PLL1VCO is described in PLL1N bits in RCU_ PLL1 register.<br>0000000: CK_PLL1P = CK_PLL1VCO<br>0000001: CK_PLL1P = CK_PLL1VCO / 2<br>0000010: CK_PLL1P = CK_PLL1VCO / 3<br>0000011: CK_PLL1P = CK_PLL1VCO / 4<br>0000100: CK_PLL1P = CK_PLL1VCO / 5<br>…<br>1111111: CK_PLL1P = CK_PLL1VCO / 128 |

| | | |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14:6 | PLL1N[8:0] | The PLL1 VCO clock multiplication factor<br>Set and reset by software (only use word / half-word write) when the PLL1 is disable. These bits used to generate PLL1 VCO clock (CK_PLL1VCO) from PLL1 VCO source clock (CK_PLL1VCOSRC). The CK_PLL1VCOSRC is described in PLLPSC bits in RCU_PLL register<br>**Note:** The frequency of CK_PLL1VCO is between 150MHz to 836MHz.<br>The value of PLL1N must : refer to the description of PLL0N value range<br>000000000: Reserved<br>…<br>000000111: Reserved<br>000001000: PLL1N = 9.<br>…<br>001000000: PLL1N = 65<br>001000001: PLL1N = 66<br>…<br>111111111: PLL1N = 512 |
| 5:0 | PLL1PSC[5:0] | The PLL1 VCO source clock prescaler<br>Set and reset by software when the PLL1 is disable. These bits used to generate the clock of PLL1 VCO source clock (CK_PLL1VCOSRC) from PLL1 source clock (CK_PLL1SRC) which is described in PLLSEL in RCU_PLLALL register.<br>The VCO source clock is between 1M to 16MHz.<br>000000: Reserved.<br>000001: CK_PLL1SRC<br>000010: CK_PLL1SRC / 2<br>000011: CK_PLL1SRC / 3<br>…<br>111111: CK_PLL1SRC / 63 |

## 6.4.33. PLL2 register (RCU_PLL2)

Address offset: 0x88

Reset value: 0x0101 2020

To configure the PLL2 clock, refer to the following formula:

CK_PLL2VCOSRC = CK_PLL2SRC / PLL2PSC

CK_PLL2VCO = CK_ PLL2VCOSRC × (PLL2N + PLL2FRAN / $2^{13}$)

CK_PLL2P = CK_ PLL2VCO / PLL2P

CK_PLL2R = CK_ PLL2VCO / PLL2R

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL2R[6:0] | | | | | | | Reserved | PLL2P[6:0] | | | | | | |
| | rw | | | | | | | | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLL2N[8:0] | | | | | | | | | PLL2PSC[5:0] | | | | | |
| | rw | | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | PLL2R[6:0] | The PLL2R output frequency division factor from PLL2 VCO clock<br>Set and reset by software when the PLL2 is disable. These bits used to generate PLL2R output clock (CK_PLL2R) from PLL2 VCO clock (CK_PLL2VCO). The CK_PLL2VCO is described in PLL2N bits in RCU_PLL2 register.<br>0000000: CK_PLL2R = CK_PLL2VCO<br>0000001: CK_PLL2R = CK_PLL2VCO / 2<br>0000010: CK_PLL2R = CK_PLL2VCO / 3<br>0000011: CK_PLL2R = CK_PLL2VCO / 4<br>0000100: CK_PLL2R = CK_PLL2VCO / 5<br>…<br>1111111: CK_PLL2R = CK_PLL2VCO / 128 |
| 23 | Reserved | Must be kept at reset value. |
| 22:16 | PLL2P[6:0] | The PLL2 P output frequency division factor from PLL2 VCO clock<br>Set and reset by software when the PLL2 is disable. These bits used to generator PLL2 P output clock (CK_PLL2P) from PLL2 VCO clock (CK_PLL2VCO). The CK_PLL2P is used to USBHS (48MHz), TRNG (48MHz). The CK_PLL2VCO is described in PLL2N bits in RCU_PLL2 register.<br>0000000: CK_PLL2P = CK_PLL2VCO<br>0000001: CK_PLL2P = CK_PLL2VCO / 2<br>0000010: CK_PLL2P = CK_PLL2VCO / 3<br>0000011: CK_PLL2P = CK_PLL2VCO / 4<br>0000100: CK_PLL2R = CK_PLL2VCO / 5<br>…<br>1111111: CK_PLL2R = CK_PLL2VCO / 128 |
| 15 | Reserved | Must be kept at reset value. |
| 14:6 | PLL2N[8:0] | The PLL2 VCO clock multiplication factor<br>Set and reset by software (only use word / half-word write) when the PLL2 is disable. These bits used to generate PLL2 VCO clock (CK_PLL2VCO) from PLL2 VCO source clock (CK_PLL2VCOSRC). The CK_PLL2VCOSRC is described in PLLPSC bits in RCU_PLL register.<br>**Note:** The frequency of CK_PLL2VCO is between 150MHz to 836MHz<br>The value of PLL2N must : refer to the description of PLL0N value range<br>000000000: Reserved<br>…<br>000000111: Reserved<br>000001000: PLL2N = 9. |

| | | |
|---|---|---|
| | | …<br>001000000: PLL2N = 65<br>001000001: PLL2N = 66<br>…<br>111111111: PLL2N = 512 |
| 5:0 | PLL2PSC[5:0] | The PLL2 VCO source clock prescaler<br>Set and reset by software when the PLL2 is disable. These bits used to generate the clock of PLL2 VCO source clock (CK_PLL2VCOSRC) from PLL2 source clock (CK_PLL2SRC) which is described in PLLSEL in RCU_PLLALL register.<br>The VCO source clock is between 1M to 16MHz.<br>000000: Reserved .<br>000001: CK_PLL2SRC<br>000010: CK_PLL2SRC / 2<br>000011: CK_PLL2SRC / 3<br>…<br>111111: CK_PLL2SRC / 63 |

## 6.4.34. Clock configuration register 1 (RCU_CFG1)

Address offset: 0x8C

Reset value: 0x0000 3F00

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HPDFSEL | | Reserved | | | | | TIMERSEL | USART5SEL[1:0] | | USART2SEL[1:0] | | USART1SEL[1:0] | | PLL2RDIV[1:0] | |
| rw | | | | | | | rw | rw | | rw | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PERSEL[1:0] | | CAN2SEL[1:0] | | CAN1SEL[1:0] | | CAN0SEL[1:0] | | Reserved | | | | | | USART0SEL[1:0] | |
| rw | | rw | | rw | | rw | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | HPDFSEL | HPDF clock source selection<br>Set and reset by software to control the HPDF clock source<br>0: CK_APB2 selected as HPDF source clock<br>1: CK_AHB selected as HPDF source clock |
| 30:25 | Reserved | Must be kept at reset value. |
| 24 | TIMERSEL | TIMER clock selection<br>This bit is set and reset by software. This bit defined all timer clock selection.<br>0: If APB1PSC / APB2PSC in RCU_CFG0 register is 0b0xx(CK_APBx = CK_AHB) or 0b100(CK_APBx = CK_AHB/2), the TIMER clock is equal to CK_AHB(CK_TIMERx = CK_AHB). Or else, the TIMER clock is twice the corresponding APB clock (TIMER in APB1 domain: CK_TIMERx = 2 x CK_APB1; |

| | | |
|---|---|---|
| | | TIMER in APB2 domain: CK_TIMERx = 2 x CK_APB2).<br>1: If APB1PSC / APB2PSC in RCU_CFG0 register is 0b0xx(CK_APBx = CK_AHB), 0b100(CK_APBx = CK_AHB / 2), or 0b101(CK_APBx = CK_AHB / 4), the TIMER clock is equal to CK_AHB(CK_TIMERx = CK_AHB). Or else, the TIMER clock is four times the corresponding APB clock (TIMER in APB1 domain: CK_TIMERx = 4 x CK_APB1, TIMER in APB2 domain: CK_TIMERx = 4 x CK_APB2). |
| 23:22 | USART5SEL[1:0] | USART5 clock source selection<br>Set and reset by software to control the USART5 clock source<br>00: CK_APB2 output clock selected as USART5 source clock<br>01: CK_AHB output clock selected as USART5 source clock<br>10: CK_LXTAL output clock selected as USART5 source clock<br>11: CK_IRC64MDIV output clock selected as USART5 source clock |
| 21:20 | USART2SEL[1:0] | USART2 clock source selection<br>Set and reset by software to control the USART2 clock source<br>00: CK_APB1 selected as USART2 source clock<br>01: CK_AHB selected as USART2 source clock<br>10: CK_LXTAL selected as USART2 source clock<br>11: CK_IRC64MDIV selected as USART2 source clock |
| 19:18 | USART1SEL[1:0] | USART1 clock source selection<br>Set and reset by software to control the USART1 clock source<br>00: CK_APB1 selected as USART1 source clock<br>01: CK_AHB selected as USART1 source clock<br>10: CK_LXTAL selected as USART1 source clock<br>11: CK_IRC64MDIV selected as USART1 source clock |
| 17:16 | PLL2RDIV[1:0] | The divider factor from PLL2R clock<br>These bits are set and reset by software when PLL2 is disabled.<br>00: CK_PLL2R / 2<br>01: CK_PLL2R / 4<br>10: CK_PLL2R / 8<br>11: CK_PLL2R / 16 |
| 15:14 | PERSEL[1:0] | CK_PER clock selection<br>Set and reset by software to control the CK_PER clock source<br>00: CK_IRC64MDIV selected as CK_PER source clock<br>01: CK_LPIRC4M selected as CK_PER source clock<br>10: CK_HXTAL selected as CK_PER source clock<br>11: reserved |
| 13:12 | CAN2SEL[1:0] | CAN2 clock source selection<br>Set and reset by software to control the CAN2 clock source<br>00: CK_HXTAL selected as CAN2 source clock<br>01: CK_APB2 selected as CAN2 source clock<br>10: CK_APB2 / 2 selected as CAN2 source clock |

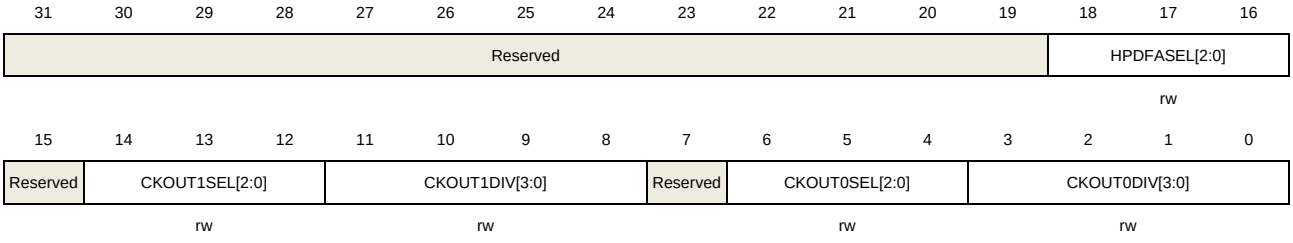
| | | |
|---|---|---|
| | | 11: CK_IRC64MDIV selected as CAN2 source clock |
| 11:10 | CAN1SEL[1:0] | CAN1 clock selection<br>Set and reset by software to control the CAN1 clock source<br>00: CK_HXTAL selected as CAN1 source clock<br>01: CK_APB2 selected as CAN1 source clock<br>10: CK_APB2 / 2 selected as CAN1 source clock<br>11: CK_IRC64MDIV selected as CAN1 source clock |
| 9:8 | CAN0SEL[1:0] | CAN0 clock source selection<br>Set and reset by software to control the CAN0 clock source<br>00: CK_HXTAL selected as CAN0 source clock<br>01: CK_APB2 selected as CAN0 source clock<br>10: CK_APB2 / 2 selected as CAN0 source clock<br>11: CK_IRC64MDIV selected as CAN0 source clock |
| 7:2 | Reserved | Must be kept at reset value. |
| 1:0 | USART0SEL[1:0] | USART0 clock source selection<br>Set and reset by software to control the USART0 clock source<br>00: CK_APB2 selected as USART0 source clock<br>01: CK_AHB selected as USART0 source clock<br>10: CK_LXTAL selected as USART0 source clock<br>11: CK_IRC64MDIV selected as USART0 source clock |

## 6.4.35. Clock configuration register 2 (RCU_CFG2)

Address offset: 0x90

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18:16 | HPDFASEL[2:0] | HPDF audio clock source selection<br>Set and reset by software to control the HPDF audio clock source<br>000: CK_PLL0Q selected as HPDF audio source clock<br>001: CK_PLL1P selected as HPDF audio source clock<br>010: CK_PLL2P selected as HPDF audio source clock<br>011: I2S_CKIN selected as HPDF audio source clock |

| | | |
|---|---|---|
| | | 100: CK_PER selected as HPDF audio source clock<br>others: Reserved |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | CKOUT1SEL[2:0] | CKOUT1 clock source selection<br>Set and reset by software.<br>000: CK_SYS clock selected<br>001: CK_PLL1R clock selected<br>010: CK_HXTAL clock selected<br>011: CK_PLL0P clock selected<br>100: CK_LPIRC4M clock selected<br>101: CK_IRC32K clock selected<br>110: CK_PLL2R clock selected<br>111: reserved<br>**Note:** Configuration of this bit field may cause interference with CK_OUT1, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs. |
| 11:8 | CKOUT1DIV[3:0] | The CK_OUT1 divider which the CK_OUT1 frequency can be reduced see bits14:12 of RCU_CFG2 for CK_OUT1<br>0000: inhibit predividers<br>0001: CK_OUT1 is divided by 1<br>0010: CK_OUT1 is divided by 2<br>0011: CK_OUT1 is divided by 3<br>0100: CK_OUT1 is divided by 4<br>…<br>1111: CK_OUT1 is divided by 15<br>**Note:** Configuration of this bit field may cause interference with CK_OUT1, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs. |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | CKOUT0SEL[2:0] | CKOUT0 clock source selection<br>Set and reset by software<br>000: CK_OUT0 selects CK_IRC64MDIV<br>001: CK_OUT0 selects CK_LXTAL<br>010: CK_OUT0 selects CK_HXTAL<br>011: CK_OUT0 selects CK_PLL0P<br>100: CK_OUT0 selects CK_IRC48M<br>101: CK_OUT0 selects CK_PER<br>110: CK_OUT0 selects USBHS0 60M<br>111: CK_OUT0 selects USBHS1 60M<br>**Note:** Configuration of this bit field may cause interference with CK_OUT0, and it is strongly recommended to configure these bits only |

| | | |
|---|---|---|
| | | after resetting but before enabling HXTAL and PLLs. |
| 3:0 | CKOUT0DIV[3:0] | The CK_OUT0 divider which the CK_OUT0 frequency can be reduced see bits 6:4 of RCU_CFG2 for CK_OUT0<br>0000: inhibit predividers<br>0001: The CK_OUT0 is divided by 1<br>0010: The CK_OUT0 is divided by 2<br>0011: The CK_OUT0 is divided by 3<br>0100: The CK_OUT0 is divided by 4<br>…<br>1111: The CK_OUT0 is divided by 15<br>**Note:** Configuration of this bit field may cause interference with CK_OUT0, and it is strongly recommended to configure these bits only after resetting but before enabling HXTAL and PLLs. |

## 6.4.36. Clock configuration register 3 (RCU_CFG3)

Address offset: 0x94

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ADC2SEL[1:0] | | ADC01SEL[1:0] | | Reserved | DSPWUSSEL | Reserved | | | | | | | |
| | | rw | | rw | | | rw | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | I2C3SEL[1:0] | | I2C2SEL[1:0] | | I2C1SEL[1:0] | |
| | | | | | | | | | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:28 | ADC2SEL[1:0] | ADC2 clock source selection<br>Set and reset by software to control the ADC2 clock source<br>00: CK_PLL1P selected as ADC2 source clock<br>01: CK_PLL2R selected as ADC2 source clock<br>10: CK_PER selected as ADC2 source clock<br>11: reserved |
| 27:26 | ADC01SEL[1:0] | ADC0 and ADC1 clock source selection<br>Set and reset by software to control the ADC0 and ADC1 clock source<br>00: CK_PLL1P selected as ADC0 and ADC1 source clock<br>01: CK_PLL2R selected as ADC0 and ADC1 source clock<br>10: CK_PER selected as ADC0 and ADC1 source clock<br>11: reserved |

| | | |
|---|---|---|
| 25 | Reserved | Must be kept at reset value. |
| 24 | DSPWUSSEL | Deep-sleep wakeup system clock source selection<br>Set and reset by software to select the system wakeup clock from deep-sleep mode.<br>The selected clock is also used as emergency clock for the Clock stuck on HXTAL.<br>0: The CK_IRC64MDIV is selected as wake up system clock from deep-sleep mode<br>1: The CK_LPIRC4M is selected as wake up system clock from deep-sleep mode<br>**Note:** If DSPWUSSEL = '1' and peripheral clock source select CK_IRC64MDIV, when system is wakeup form deep-sleep mode by this peripheral, if peripheral wakeup function turns off, the peripheral will have no clock because of IRC64M is turn off. In this case, user needs to set IRC64MEN in RCU_CTL register to turn on IRC64M clock again. When the CKMEN bit is set and the system clock is CK_HXTAL or switched to HXTAL, this bit cannot be changed. |
| 23:6 | Reserved | Must be kept at reset value. |
| 5:4 | I2C3SEL[1:0] | I2C3 clock source selection<br>Set and reset by software to control the I2C3 clock source.<br>00: CK_APB1 selected as I2C3 source clock<br>01: CK_PLL2R selected as I2C3 source clock<br>10: CK_IRC64MDIV selected as I2C3 source clock<br>11: CK_LPIRC4M selected as I2C3 source clock |
| 3:2 | I2C2SEL[1:0] | I2C2 clock source selection<br>Set and reset by software to control the I2C2 clock source.<br>00: CK_APB1 selected as I2C2 source clock<br>01: CK_PLL2R selected as I2C2 source clock<br>10: CK_IRC64MDIV selected as I2C2 source clock<br>11: CK_LPIRC4M selected as I2C2 source clock |
| 1:0 | I2C1SEL[1:0] | I2C1 clock source selection<br>Set and reset by software to control the I2C1 clock source.<br>00: CK_APB1 selected as I2C1 source clock<br>01: CK_PLL2R selected as I2C1 source clock<br>10: CK_IRC64MDIV selected as I2C1 source clock<br>11: CK_LPIRC4M selected as I2C1 source clock |

### 6.4.37. PLL all configuration register (RCU_PLLALL)

Address offset: 0x98

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | PLLSEL[1:0] | |
| | | | | | | | | | | | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | PLL2VCOSEL | PLL2RNG[1:0] | | Reserved | PLL1VCOSEL | PLL1RNG[1:0] | | Reserved | PLL0VCOSEL | PLL0RNG[1:0] | |
| | | | | | rw | rw | | | rw | rw | | | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17:16 | PLLSEL[1:0] | PLLs clock source selection<br>Set and reset by software to control the PLLs clock source.<br>00: CK_IRC64MDIV selected as source clock of PLL0, PLL1, PLL2<br>01: CK_LPIRC4M selected as source clock of PLL0, PLL1, PLL2<br>10: CK_HXTAL slected as source clock of PLL0, PLL1, PLL2<br>11: No clock selected as source clock of PLL0, PLL1, PLL2 |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | PLL2VCOSEL | PLL2 VCO selection<br>Set and reset by software when the PLL2 is disable.<br>0: select wide VCO, range: 192 - 836MHz , PLL2 input clock frequency range should be 2 -16 MHz<br>1: select narrow VCO, range: 150 - 420MHz , PLL2 input clock frequency range should be 1 -2 MHz |
| 9:8 | PLL2RNG[1:0] | PLL2 input clock range<br>Set and reset by software when the PLL2 is disable.<br>00: input clock frequency: 1 - 2MHz<br>01: input clock frequency: 2 - 4MHz<br>10: input clock frequency: 4 - 8MHz<br>11: input clock frequency: 8 - 16MHz |
| 7 | Reserved | Must be kept at reset value. |
| 6 | PLL1VCOSEL | PLL1 VCO selection<br>Set and reset by software when the PLL1 is disable.<br>0: select wide VCO, range: 192 - 836MHz , PLL1 input clock frequency range should be 2 -16 MHz<br>1: select narrow VCO, range: 150 - 420MHz , PLL1 input clock frequency range should be 1 -2 MHz |
| 5:4 | PLL1RNG[1:0] | PLL1 input clock range<br>Set and reset by software when the PLL1 is disable.<br>00: input clock frequency: 1 - 2MHz<br>01: input clock frequency: 2 - 4MHz |

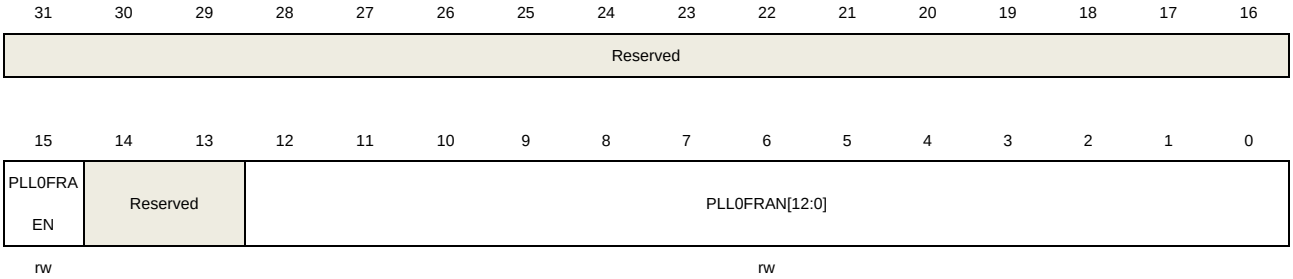
| | | |
|---|---|---|
| | | 10: input clock frequency: 4 - 8MHz<br>11: input clock frequency: 8 - 16MHz |
| 3 | Reserved | Must be kept at reset value. |
| 2 | PLL0VCOSEL | PLL0 VCO selection<br>Set and reset by software when the PLL0 is disable.<br>0: select wide VCO, range: 192 - 836MHz , PLL0 input clock frequency range should be 2 -16 MHz<br>1: select narrow VCO, range: 150 - 420MHz , PLL0 input clock frequency range should be 1 -2 MHz |
| 1:0 | PLL0RNG[1:0] | PLL0 input clock range<br>Set and reset by software when the PLL0 is disable..<br>00: input clock frequency: 1 - 2MHz<br>01: input clock frequency: 2 - 4MHz<br>10: input clock frequency: 4 - 8MHz<br>11: input clock frequency: 8 - 16MHz |

## 6.4.38. PLL0 fraction configuration register (RCU_PLL0FRA)

Address offset: 0x9C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



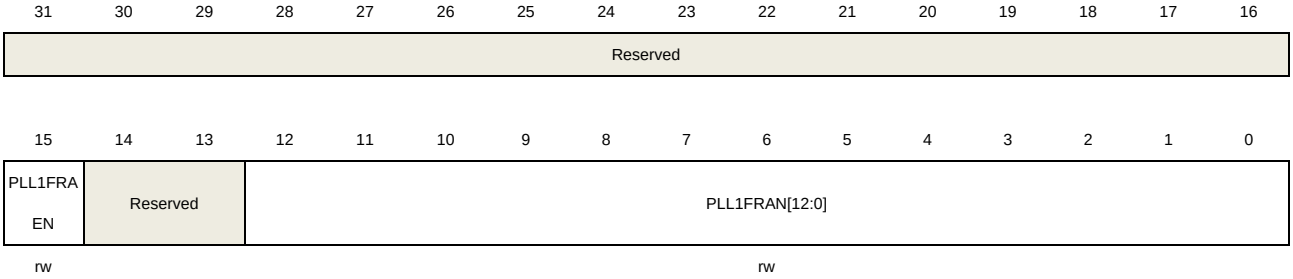

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | PLL0FRAEN | PLL0 fractional latch enable.<br>These bits are set and reset by software. These bits can lock the PLL0FRAN value into a Sigma-Delta modulator .When PLL0FRAEN bit switchs from "0" to "1", the PLL0FRAN value will transfer to modulator. |
| 14:13 | Reserved | Must be kept at reset value. |
| 12:0 | PLL0FRAN[12:0] | Fractional part of the multiplication factor for PLL0 VCO<br>These bits are set and reset by software. These bits can control the fractional part of the multiplication for PLL0 VCO. This bit field can be modified dynamically to fine-tune the PLL0 VCO.<br>These bits must configure the PLL0 VCO out frequency to the following range: |

When PLL0VCOSEL = 0, the range is 192MHz to 836MHz; When PLL0VCOSEL = 1, the range is 150MHz to 420MHz.

## 6.4.39. PLL1 fraction configuration register (RCU_PLL1FRA)

Address offset: 0xA0
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

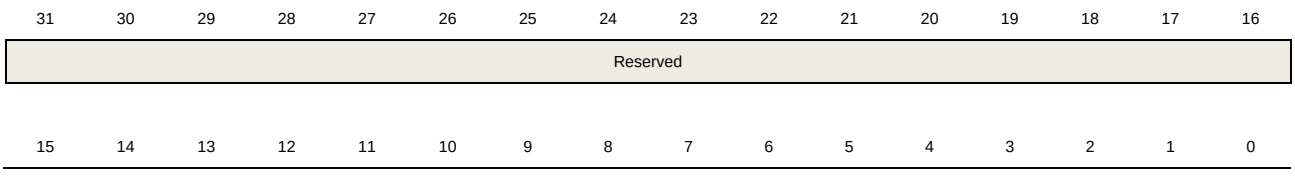



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | PLL1FRAEN | PLL1 fractional latch enable.<br>These bits are set and reset by software. These bits can lock the PLL1FRAN value into a Sigma-Delta modulator .When PLL1FRAEN bit switchs from "0" to "1", the PLL1FRAN value will transfer to modulator. |
| 14:13 | Reserved | Must be kept at reset value. |
| 12:0 | PLL1FRAN [12:0] | Fractional part of the multiplication factor for PLL1 VCO<br>These bits are set and reset by software. These bits can control the fractional part of the multiplication for PLL1 VCO. This bit field can be modified dynamically to fine-tune the PLL1 VCO.<br>These bits must configure the PLL1 VCO out frequency to the following range:<br>When PLL1VCOSEL = 0, the range is 192MHz to 836MHz; When PLL1VCOSEL = 1, the range is 150MHz to 420MHz. |

## 6.4.40. PLL2 fraction configuration register (RCU_PLL2FRA)

Address offset: 0xA4
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



| PLL2FRAEN | Reserved | PLL2FRAN[12:0] |
|---|---|---|
| rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | PLL2FRAEN | PLL2 fractional latch enable.<br>These bits are set and reset by software. These bits can lock the PLL2FRAN value into a Sigma-Delta modulator .When PLL2FRAEN bit switchs from "0" to "1", the PLL2FRAN value will transfer to modulator. |
| 14:13 | Reserved | Must be kept at reset value. |
| 12:0 | PLL2FRAN[12:0] | Fractional part of the multiplication factor for PLL2 VCO<br>These bits are set and reset by software. These bits can control the fractional part of the multiplication for PLL2 VCO. This bit field can be modified dynamically to fine-tune the PLL2 VCO.<br>These bits must configure the PLL2 VCO out frequency to the following range: When PLL2VCOSEL = 0, the range is 192MHz to 836MHz; When PLL2VCOSEL = 1, the range is 150MHz to 420MHz. |

## 6.4.41. Additional clock control register 0 (RCU_ADDCTL0)

Address offset: 0xC0

Reset value: 0x8000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IRC48MCALIB[7:0] | | | | | | | | Reserved | | | | | | IRC48MSTB | IRC48MEN |
| r | | | | | | | | | | | | | | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | PLL48MSEL | CK48MSEL |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | IRC48MCALIB [7:0] | Internal 48MHz RC oscillator calibration value register<br>These bits are load automatically at power on. |
| 23:18 | Reserved | Must be kept at reset value. |
| 17 | IRC48MSTB | Internal 48MHz RC oscillator clock stabilization Flag<br>Set by hardware to indicate if the IRC48M oscillator is stable and ready for use.<br>0: IRC48M is not stable |

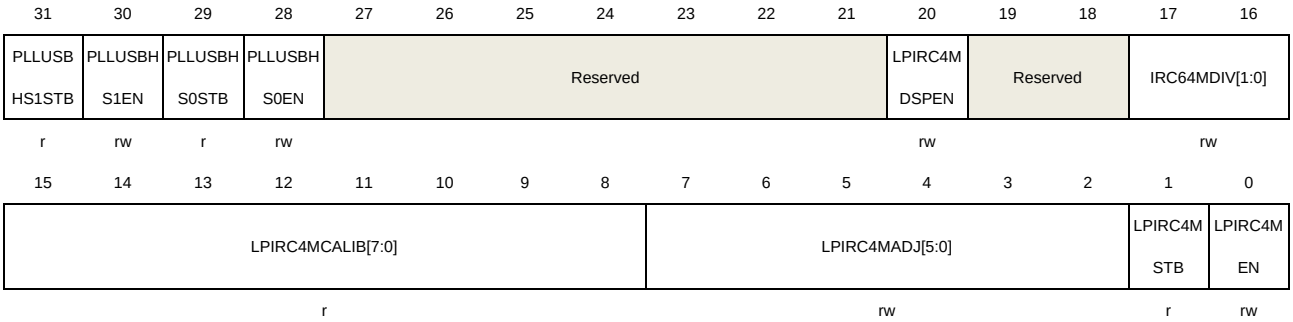
| | | |
|---|---|---|
| | | 1: IRC48M is stable |
| 16 | IRC48MEN | Internal 48MHz RC oscillator enable<br>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.<br>0: IRC48M disable<br>1: IRC48M enable |
| 15:2 | Reserved | Must be kept at reset value. |
| 1 | PLL48MSEL | PLL48M clock selection<br>Set and reset by software. This bit used to generate PLL48M clock which select CK_PLL0Q or CK_PLL2P clock.<br>0: Select CK_PLL0Q clock<br>1: Select CK_PLL2P clock |
| 0 | CK48MSEL | 48MHz clock selection<br>Set and reset by software. This bit used to generate CK48M clock which select IRC48M clock or PLL48M clock. The CK48M clock used for TRNG/USBHS. The PLL48M clock refer to PLL48MSEL bit in RCU_ADDCTL register.<br>0: Don't select IRC48M clock (use CK_PLL0Q clock or CK_PLL2P clock select by PLL48MSEL)<br>1: Select IRC48M clock |

## 6.4.42. Additional clock control register 1(RCU_ADDCTL1)

Address offset: 0xC4

Reset value: 0x0000 7080

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PLLUSBHS1STB | PLLUSBHS1 clock stabilization flag<br>Set by hardware to indicate if the PLLUSBHS1 clock is stable and ready for use<br>0: PLLUSBHS1 clock is not stable<br>1: PLLUSBHS1 clock is stabilized |
| 30 | PLLUSBHS1EN | PLLUSBHS1 clock enable<br>Set and reset by software. Reset by hardware when enter deep-sleep or standby |

| | | |
|---|---|---|
| | | mode.<br>0: Disabled PLLUSBHS1 clock<br>1: Enabled PLLUSBHS1 clock |
| 29 | PLLUSBHS0STB | PLLUSBHS0 clock stabilization flag<br>Set by hardware to indicate if the PLLUSBHS0 clock is stable and ready for use<br>0: PLLUSBHS0 clock is not stable<br>1: PLLUSBHS0 clock is stabilized |
| 28 | PLLUSBHS0EN | PLLUSBHS0 clock enable<br>Set and reset by software. Reset by hardware when enter deep-sleep or standby mode.<br>0: Disabled PLLUSBHS0 clock<br>1: Enabled PLLUSBHS0 clock |
| 27:21 | Reserved | Must be kept at reset value. |
| 20 | LPIRC4MDSPEN | LPIRC4M clock enable in deepsleep mode<br>Set and reset by software. LPIRC4M can be forced on even in deepsleep mode to quickly be used as a kernel clock for some peripherals. This bit has no effect on the value of LPIRC4MEN.<br>0: Has no impact on LPIRC4M<br>1: Force LPIRC4M to run even in deepsleep mode |
| 19:18 | Reserved | Must be kept at reset value. |
| 17:16 | IRC64MDIV[1:0] | IRC64M clock divider<br>Set and reset by software. It cannot be written when system clock select CK_IRC64MDIV or IRC64MEN is set.<br>00: CK_IRC64MDIV = CK_IRC64M / 1<br>01: CK_IRC64MDIV = CK_IRC64M / 2<br>10: CK_IRC64MDIV = CK_IRC64M / 4<br>11: CK_IRC64MDIV = CK_IRC64M / 8 |
| 15:8 | LPIRC4MCALIB[7:0] | LPIRC4M calibration value<br>These bits are load automatically at power on.The calibration signal step is 0.4%. |
| 7:2 | LPIRC4MADJ[5:0] | LPIRC4M frequency clock trim adjust value<br>These bits are set by software. The trimming value is these bits (LPIRC4MADJ[) added to the LPIRC4MCALIB[7:0] bits. The trimming value should trim the LPIRC4MCALIB to 4 MHz ± 1%. |
| 1 | LPIRC4MSTB | LPIRC4M clock stabilization flag<br>Set by hardware to indicate if the LPIRC4M oscillator is stable and ready for use.<br>0: LPIRC4M RC oscillator is not stable<br>1: LPIRC4M RC oscillator is stabilized |
| 0 | LPIRC4MEN | LPIRC4M clock enable<br>Set and reset by software. This bit cannot be reset if the LPIRC4M clock is used |

as the system clock. If DSPWUSSEL = 1, set by hardware when leaving Deep-sleep mode.

0: LPIRC4M RC oscillator disabled

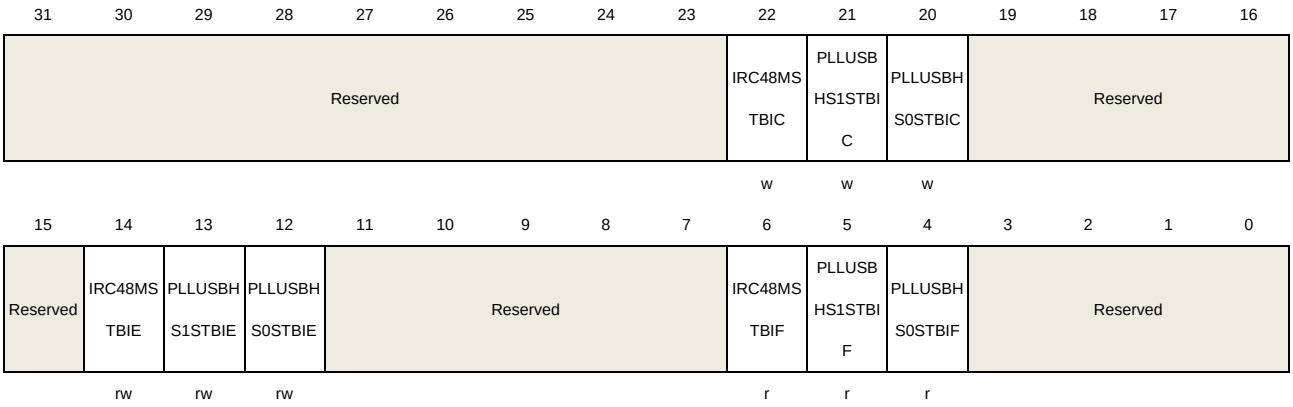
1: LPIRC4M RC oscillator enabled

## 6.4.43. Additional clock interrupt register (RCU_ADDINT)

Address offset: 0xCC

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | IRC48MSTBIC | Internal 48 MHz RC oscillator Stabilization interrupt clear<br>Write 1 by software to reset the IRC48MSTBIF flag.<br>0: Not reset IRC48MSTBIF flag<br>1: Reset IRC48MSTBIF flag |
| 21 | PLLUSBHS1STBIC | Internal PLL of USBHS1 Stabilization interrupt clear<br>Write 1 by software to reset the PLLUSBHS1STBIF flag.<br>0: Not reset PLLUSBHS1STBIF flag<br>1: Reset PLLUSBHS1STBIF flag |
| 20 | PLLUSBHS0STBIC | Internal PLL of USBHS0 Stabilization interrupt clear<br>Write 1 by software to reset the PLLUSBHS0STBIF flag.<br>0: Not reset PLLUSBHS0STBIF flag<br>1: Reset PLLUSBHS0STBIF flag |
| 19:15 | Reserved | Must be kept at reset value. |
| 14 | IRC48MSTBIE | Internal 48 MHz RC oscillator Stabilization interrupt enable<br>Set and reset by software to enable/disable the IRC48M stabilization interrupt<br>0: Disable the IRC48M stabilization interrupt<br>1: Enable the IRC48M stabilization interrupt |

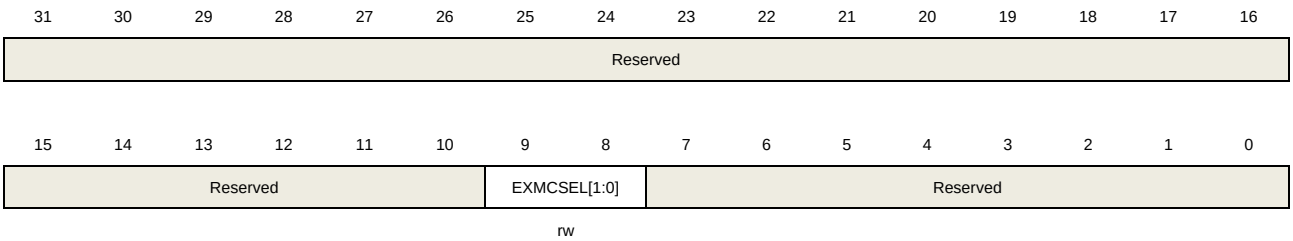
| | | |
|---|---|---|
| 13 | PLLUSBHS1STBIE | Internal PLL of USBHS1 Stabilization interrupt enable<br>Set and reset by software to enable/disable the USBHS1 PLL stabilization interrupt<br>0: Disable the USBHS1 PLL stabilization interrupt<br>1: Enable the USBHS1 PLL stabilization interrupt |
| 12 | PLLUSBHS0STBIE | Internal PLL of USBHS0 Stabilization interrupt enable<br>Set and reset by software to enable/disable the USBHS0 PLL stabilization interrupt<br>0: Disable the USBHS0 PLL stabilization interrupt<br>1: Enable the USBHS0 PLL stabilization interrupt |
| 11:7 | Reserved | Must be kept at reset value. |
| 6 | IRC48MSTBIF | IRC48M stabilization interrupt flag<br>Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set.<br>Reset by software when setting the IRC48MSTBIC bit.<br>0: No IRC48M stabilization interrupt generated<br>1: IRC48M stabilization interrupt generated |
| 5 | PLLUSBHS1STBIF | Internal PLL of USBHS1 stabilization interrupt flag<br>Set by hardware when the USBHS1 PLL clock is stable and the PLLUSBHS1STBIE bit is set.<br>Reset by software when setting the PLLUSBHS1STBIC bit.<br>0: No USBHS1 PLL clock stabilization interrupt generated<br>1: USBHS1 PLL clock stabilization interrupt generated |
| 4 | PLLUSBHS0STBIF | Internal PLL of USBHS0 stabilization interrupt flag<br>Set by hardware when the USBHS0 PLL clock is stable and the PLLUSBHS0STBIE bit is set.<br>Reset by software when setting the PLLUSBHS0STBIC bit.<br>0: No USBHS0 PLL clock stabilization interrupt generated<br>1: USBHS0 PLL clock stabilization interrupt generated |
| 3:0 | Reserved | Must be kept at reset value. |

## 6.4.44. Clock configuration register 4 (RCU_CFG4)

Address offset: 0xD0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9:8 | EXMCSEL[1:0] | EXMC clock source selection<br>Set and reset by software to control the EXMC clock source<br>00: CK_AHB selected as EXMC source clock<br>01: CK_PLL0Q selected as EXMC source clock<br>10: CK_PLL1R selected as EXMC source clock<br>11: CK_PER selected as EXMC source clock |
| 7:0 | Reserved | Must be kept at reset value. |

## 6.4.45. USB clock control register (RCU_USBCLKCTL)

Address offset: 0xD4

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | USBHS1PSC[2:0] | | | USBHS0PSC[2:0] | | |
| | | | | | | | | | | | rw | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | USBHS148MSEL[1:0] | | USBHS1SWEN | PLLUSBHS1PRESEL | Reserved | USBHS1SEL | Reserved | | USBHS048MSEL[1:0] | | USBHS0SWEN | PLLUSBHS0PRESEL | Reserved | USBHS0SEL | Reserved |
| | rw | | rw | rw | | rw | | | rw | | rw | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:19 | USBHS1PSC[2:0] | USBHS1 clock prescaler selection<br>Set and reset by software to control the USBHS1 clock prescaler value. The USBHS1 clock must be 48MHz. These bits can't be reset if the USBHS1 clock is enabled.<br>000: CK_USBHS1 = CK_PLL1Q / 1<br>001: CK_USBHS1 = CK_PLL1Q / 2<br>010: CK_USBHS1 = CK_PLL1Q / 3<br>011: CK_USBHS1 = CK_PLL1Q / 4<br>100: CK_USBHS1 = CK_PLL1Q / 5<br>101: CK_USBHS1 = CK_PLL1Q / 6<br>110 :CK_USBHS1 = CK_PLL1Q / 7<br>111 :CK_USBHS1 = CK_PLL1Q / 8 |
| 18:16 | USBHS0PSC[2:0] | USBHS0 clock prescaler selection<br>Set and reset by software to control the USBHS0 clock prescaler value. The USBHS0 clock must be 48MHz. These bits can't be reset if the USBHS0 clock is |

| | | |
|---|---|---|
| | | enabled.<br>000: CK_USBHS0 = CK_PLL1Q / 1<br>001: CK_USBHS0 = CK_PLL1Q / 2<br>010: CK_USBHS0 = CK_PLL1Q / 3<br>011: CK_USBHS0 = CK_PLL1Q / 4<br>100: CK_USBHS0 = CK_PLL1Q / 5<br>101: CK_USBHS0 = CK_PLL1Q / 6<br>110 :CK_USBHS0 = CK_PLL1Q / 7<br>111 :CK_USBHS0 = CK_PLL1Q / 8 |
| 15 | Reserved | Must be kept at reset value. |
| 14:13 | USBHS148MSEL[1:0] | USBHS1 48M clock source selection<br>Set and reset by software.<br>00: CK_PLL0R selected as USBHS1 48M source clock<br>01: CK_PLLUSBHS1/USBHS1DV output clock selected as USBHS1 48M source clock<br>10: CK_PLL1Q/USBHS1PSC output selected as USBHS1 48M source clock<br>11: CK_IRC48M selected as USBHS1 48M source clock |
| 12 | USBHS1SWEN | USBHS1 clock source selection enable<br>0: Hardware switch USBHS1 clock by USBHS1 module<br>1: Use USBHS1SW to switch USBHS1 clock |
| 11 | PLLUSBHS1PRESEL | PLLUSBHS1 clock source preselection<br>Set and reset by software.<br>0: CK_HATAL selected as PLLUSBHS1 source clock<br>1: CK_IRC48M selected as PLLUSBHS1 source clock |
| 10 | Reserved | Must be kept at reset value. |
| 9 | USBHS1SEL | USBHS1 clock source selection<br>Set and reset by software.<br>0: 48M selected as USBHS1 source clock<br>1: 60M selected as USBHS1 source clock |
| 8:7 | Reserved | Must be kept at reset value. |
| 6:5 | USBHS048MSEL[1:0] | USBHS0 48M clock source selection<br>Set and reset by software.<br>00: CK_PLL0R selected as USBHS0 48M source clock<br>01: CK_PLLUSBHS0/USBHS0DV selected as USBHS0 48M source clock<br>10: CK_PLL1Q/USBHS0PSC selected as USBHS0 48M source clock<br>11: CK_IRC48M selected as USBHS0 48M source clock |
| 4 | USBHS0SWEN | USBHS0 clock source selection enable<br>0: Hardware switch USBHS0 clock by USBHS0 module<br>1: Use USBHS0SW to switch USBHS0 clock |

| | | |
|---|---|---|
| 3 | PLLUSBHS0PRESEL | PLLUSBHS0 clock source preselection<br>Set and reset by software.<br>0: CK_HATAL selected as PLLUSBHS0 source clock<br>1: CK_IRC48M selected as PLLUSBHS0 source clock |
| 2 | Reserved | Must be kept at reset value. |
| 1 | USBHS0SEL | USBHS0 clock source selection<br>Set and reset by software.<br>0: 48M selected as USBHS0 source clock<br>1: 60M selected as USBHS0 source clock |
| 0 | Reserved | Must be kept at reset value. |

## 6.4.46. PLLUSB configuration register (RCU_PLLUSBCFG)

Address offset: 0xD8

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLLUSBHS1MF[6:0] | | | | | | | Reserved | USBHS1DV[2:0] | | | PLLUSBHS1PREDV[3:0] | | | |
| | rw | | | | | | | | rw | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PLLUSBHS0MF[6:0] | | | | | | | Reserved | USBHS0DV[2:0] | | | PLLUSBHS0PREDV[3:0] | | | |
| | rw | | | | | | | | rw | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | PLLUSBHS1MF[6:0] | The PLLUSBHS1 clock multiplication factor<br>0000000: Reserved<br>0000001: Reserved<br>…<br>0001111: Reserved<br>0010000: PLLUSBHS1MF input source clock multiplied by 16<br>0010000: PLLUSBHS1MF input source clock multiplied by 17<br>0010010: PLLUSBHS1MF input source clock multiplied by 18<br>0010011: PLLUSBHS1MF input source clock multiplied by 19<br>…<br>1111111: PLLUSBHS1MF input source clock multiplied by 127<br>**Note:** The frequency of PLLUSBHS1 output clock is no more than 480MHz |
| 23 | Reserved | Must be kept at reset value. |
| 22:20 | USBHS1DV[2:0] | USBHS1 clock divider factor<br>These bits are set and reset by software. |

| | | |
|---|---|---|
| | | 000: USBHS1DV input source clock divided by 2<br>001: USBHS1DV input source clock divided by 4<br>010: USBHS1DV input source clock divided by 6<br>…<br>111: USBHS1DV input source clock divided by 16 |
| 19:16 | PLLUSBHS1PREDV[3:0] | PLLUSBHS1PREDV clock divide factor<br>These bits are set and reset by software.<br>0000: Reserved<br>0001: PLLUSBHS1PREDV input source clock divided by 1<br>0010: PLLUSBHS1PREDV input source clock divided by / 2<br>…<br>1111: PLLUSBHS1PREDV input source clock divided by / 15 |
| 15 | Reserved | Must be kept at reset value. |
| 14:8 | PLLUSBHS0MF[6:0] | The PLLUSBHS0 clock multiplication factor<br>0000000: Reserved<br>0000001: Reserved<br>…<br>0001111: Reserved<br>0010000: PLLUSBHS0MF input source clock multiplied by 16<br>0010000: PLLUSBHS0MF input source clock multiplied by 17<br>0010010: PLLUSBHS0MF input source clock multiplied by 18<br>0010011: PLLUSBHS0MF input source clock multiplied by 19<br>…<br>1111111: PLLUSBHS0MF input source clock multiplied by 127<br>**Note:** The frequency of PLLUSBHS0 output clock is no more than 480MHz |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | USBHS0DV[2:0] | USBHS0 clock divider factor<br>These bits are set and reset by software.<br>000: USBHS0DV input source clock divided by 2<br>001: USBHS0DV input source clock divided by 4<br>010: USBHS0DV input source clock divided by 6<br>…<br>111: USBHS0DV input source clock divided by 16 |
| 3:0 | PLLUSBHS0PREDV[3:0] | PLLUSBHS0PREDV clock divide factor<br>These bits are set and reset by software.<br>0000: Reserved<br>0001: PLLUSBHS0PREDV input source clock divided by 1<br>0010: PLLUSBHS0PREDV input source clock divided by / 2<br>…<br>1111: PLLUSBHS0PREDV input source clock divided by / 15 |

### 6.4.47. APB2 additional reset register (RCU_ADDAPB2RST)

Address offset: 0xE0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | CAN2RST | CAN1RST | CAN0RST |
| | | | | | | | | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | CAN2RST | CAN2 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN2 unit |
| 1 | CAN1RST | CAN1 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN1 |
| 0 | CAN0RST | CAN0 reset<br>This bit is set and reset by software.<br>0: No reset<br>1: Reset CAN0 |

### 6.4.48. APB2 additional enable register (RCU_ADDAPB2EN)

Address offset: 0xE4

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | CAN2EN | CAN1EN | CAN0EN |
| | | | | | | | | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|

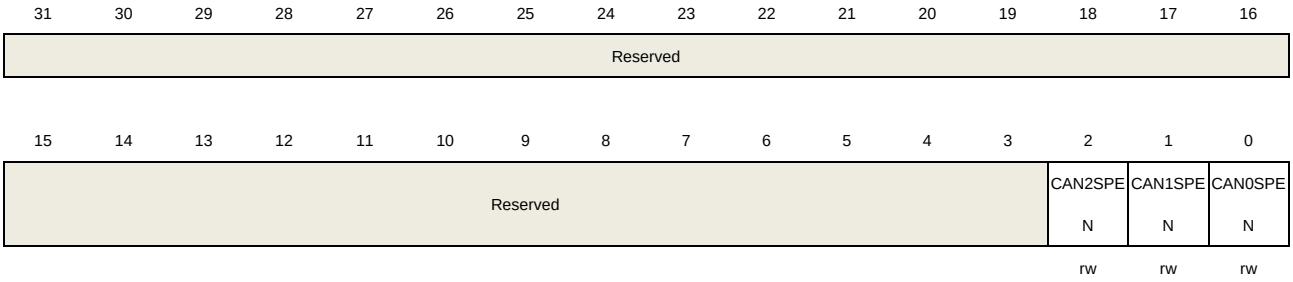
| | | |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | CAN2EN | CAN2 clock enable<br>This bit is set and reset by software.<br>0: Disable CAN2 clock<br>1: Enable CAN2 clock |
| 1 | CAN1EN | CAN1 clock enable<br>This bit is set and reset by software.<br>0: Disable CAN1 clock<br>1: Enable CAN1 clock |
| 0 | CAN0EN | CAN0 clock enable<br>This bit is set and reset by software.<br>0: Disable CAN0 clock<br>1: Enable CAN0 clock |

### 6.4.49. APB2 additional sleep enable register (RCU_ADDAPB2SPEN)

Address offset: 0xE8

Reset value: 0x0000 0007

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 30:3 | Reserved | Must be kept at reset value. |
| 2 | CAN2SPEN | CAN2 clock enable in sleep mode<br>This bit is set and reset by software.<br>0: Disable CAN2 clock in sleep mode<br>1: Enable CAN2 clock in sleep mode |
| 1 | CAN1SPEN | CAN1 clock enable in sleep mode<br>This bit is set and reset by software.<br>0: Disable CAN1 clock in sleep mode<br>1: Enable CAN1 clock in sleep mode |
| 0 | CAN0SPEN | CAN0 clock enable in sleep mode<br>This bit is set and reset by software.<br>0: Disable CAN0 clock in sleep mode |

1: Enable CAN0 clock in sleep mode

## 6.4.50. Clock configuration register 5 (RCU_CFG5)

Address offset: 0xF0

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | SPI5SEL[2:0] | | | Reserved | SPI4SEL[2:0] | | |
| | | | | | | | | | | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SPI3SEL[2:0] | | | Reserved | SPI2SEL[2:0] | | | Reserved | SPI1SEL[2:0] | | | Reserved | SPI0SEL[2:0] | | |
| | rw | | | | rw | | | | rw | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22:20 | SPI5SEL[2:0] | SPI5 / I2S5 clock source selection<br>Set and reset by software to control the SPI5 / I2S5 clock source<br>000: CK_APB2 selected as SPI5 / I2S5 source clock<br>001: CK_PLL1Q selected as SPI5 / I2S5 source clock<br>010: CK_PLL2Q selected as SPI5 / I2S5 source clock<br>011: CK_IRC64MDIV selected as SPI5 / I2S5 source clock<br>100: CK_LPIRC4M selected as SPI5 / I2S5 source clock<br>101: CK_HXTAL selected as SPI5 / I2S5 source clock<br>110: I2S_CKIN selected as SPI5 / I2S5 source clock<br>111: reserved |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | SPI4SEL[2:0] | SPI4 clock source selection<br>Set and reset by software to control the SPI4 clock source<br>000: CK_APB2 selected as SPI4 source clock<br>001: CK_PLL1Q selected as SPI4 source clock<br>010: CK_PLL2Q selected as SPI4 source clock<br>011: CK_IRC64MDIV selected as SPI4 source clock<br>100: CK_LPIRC4M selected as SPI4 source clock<br>101: CK_HXTAL selected as SPI4 source clock<br>others: reserved |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | SPI3SEL[2:0] | SPI3 clock source selection<br>Set and reset by software to control the SPI3 clock source<br>000: CK_APB2 selected as SPI3 source clock<br>001: CK_PLL1Q selected as SPI3 source clock |

| | | |
|---|---|---|
| | | 010: CK_PLL2Q selected as SPI3 source clock<br>011: CK_IRC64MDIV selected as SPI3 source clock<br>100: CK_LPIRC4M selected as SPI3 source clock<br>101: CK_HXTAL selected as SPI3 source clock<br>others: reserved |
| 11 | Reserved | Must be kept at reset value. |
| 10:8 | SPI2SEL[2:0] | SPI2 / I2S2 clock source selection<br>Set and reset by software to control the SPI2 / I2S2 clock source<br>000: CK_PLL0Q selected as SPI2 / I2S2 source clock<br>001: CK_PLL1P selected as SPI2 / I2S2 source clock<br>010: CK_PLL2P selected as SPI2 / I2S2 source clock<br>011: I2S_CKIN selected as SPI2 / I2S2 source clock<br>100: CK_PER selected as SPI2 / I2S2 source clock<br>others: reserved |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | SPI1SEL[2:0] | SPI1 / I2S1 clock source selection<br>Set and reset by software to control the SPI1 / I2S1 clock source<br>000: CK_PLL0Q selected as SPI1 / I2S1 source clock<br>001: CK_PLL1P selected as SPI1 / I2S1 source clock<br>010: CK_PLL2P selected as SPI1 / I2S1 source clock<br>011: I2S_CKIN selected as SPI1 / I2S1 source clock<br>100: CK_PER selected as SPI1 / I2S1 source clock<br>others: reserved |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | SPI0SEL[2:0] | SPI0 / I2S0 clock source selection<br>Set and reset by software to control the SPI0 / I2S0 clock source<br>000: CK_PLL0Q selected as SPI0 / I2S0 source clock<br>001: CK_PLL1P selected as SPI0 / I2S0 source clock<br>010: CK_PLL2P selected as SPI0 / I2S0 source clock<br>011: I2S_CKIN selected as SPI0 / I2S0 source clock<br>100: CK_PER selected as SPI0 / I2S0 source clock<br>others: reserved |

# 7. Clock trim controller (CTC)

## 7.1. Overview

The Clock Trim Controller (CTC) is used to trim internal 48MHz RC oscillator (IRC48M) automatically by hardware. The CTC unit trim the frequency of the IRC48M based on an external accurate reference signal source. It can automatically adjust the trim value to provide a precise IRC48M clock.

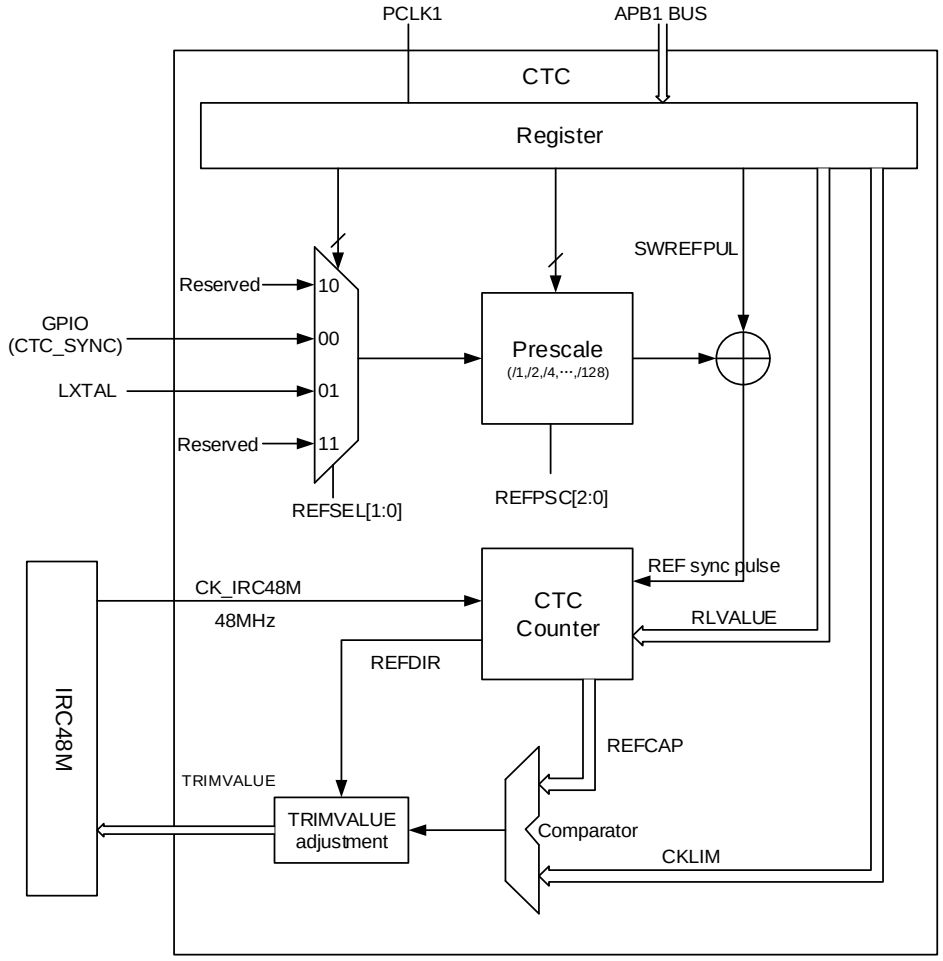
## 7.2. Characteristics

- Two external reference signal sources: GPIO(CTC_SYNC) and LXTAL clock.
- Provide software reference sync pulse.
- Automatically trimmed by hardware without any software action.
- 16 bits trim counter with reference signal source capture and reload.
- 8 bits clock trim base value to frequency evaluation and automatically trim.
- Enough flag or interrupt to indicate the clock is OK (CKOKIF), warning (CKWARNIF) or error (ERRIF).

## 7.3. Function overview

***Figure 7-1. CTC overview*** provides details on the internal configuration of the CTC.

**Figure 7-1. CTC overview**




### 7.3.1. REF sync pulse generator

Firstly, the reference signal source can select GPIO(CTC_SYNC), or LXTAL clock by setting REFSEL bits in CTC_CTL1 register.

Secondly, the selected reference signal source use a configurable polarity by setting REFPOL bit in CTC_CTL1 register, and can be divided to a suitable frequency with a configurable prescaler by setting REFPSC bits in CTC_CTL1 register.

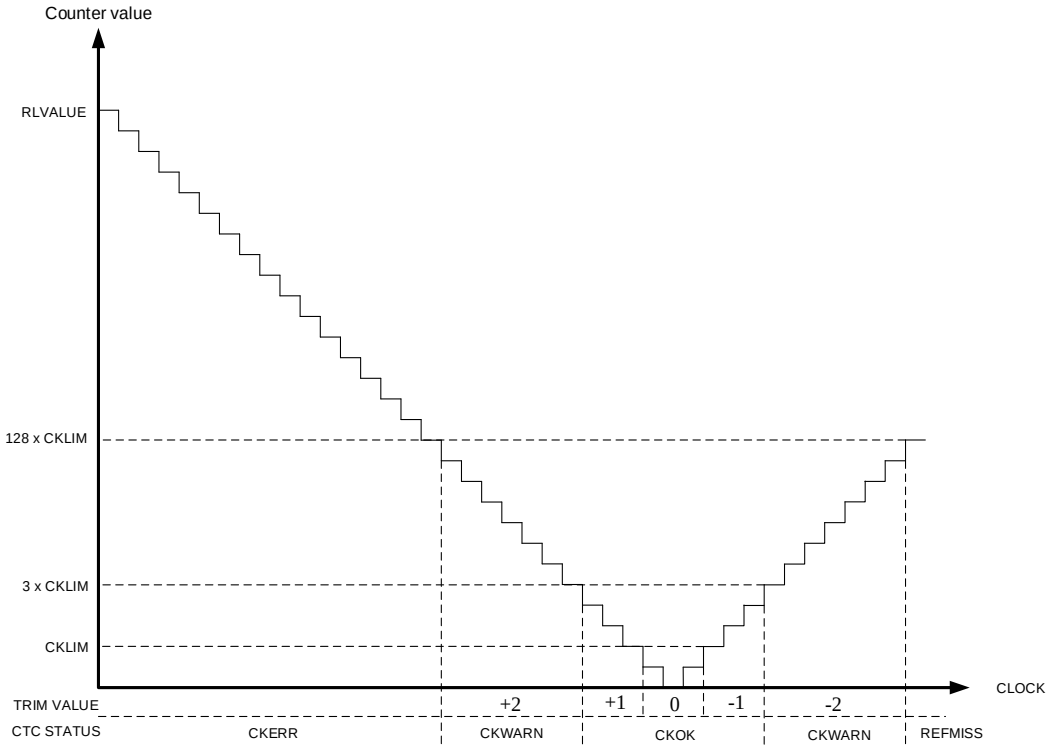
Thirdly, if a software reference pulse needed, write 1 to SWREFPUL bit in CTC_CTL0 register. The software reference pulse generated in last step is logical OR with the external reference pulse.

### 7.3.2. CTC trim counter

The CTC trim counter is clocked by CK_IRC48M. After CNTEN bit in CTC_CTL0 register set, and a first REF sync pulse detected, the counter start down-counting from RLVALUE (defined in CTC_CTL1 register). If any REF sync pulse detected, the counter reload the RLVALUE and start down-counting again. If no REF sync pulse detected, the counter down-count to zero,

and then up- counting to 128 x CKLIM (defined in CTC_CTL1 register), and then stop until next REF sync pulse detected. If any REF sync pulse detected, the current CTC trim counter value is captured to REFCAP in status register (CTC_STAT), and the counter direction is captured to REFDIR in status register (CTC_STAT). The detail is showing in ***Figure 7-2. CTC trim counter***.

**Figure 7-2. CTC trim counter**




### 7.3.3. Frequency evaluation and automatically trim process

The clock frequency evaluation is performed when a REF sync pulse occur. If a REF sync pulse occurs on down-counting, it means the current clock is slower than correct clock (the frequency of 48M).It needs to improve TRIMVALUE in CTC_CTL0 register. If a REF sync pulse occurs on up-counting, it means the current clock is faster than correct clock (the frequency of 48M).It needs to reduce TRIMVALUE in CTC_CTL0 register. The CKOKIF, CKWARNIF, CKERR and REFMISS in CTC_STAT register shows the frequency evaluation scope.

If the AUTOTRIM bit in CTC_CTL0 register is setting, the automatically hardware trim mode enabled. In this mode, if a REF sync pulse occurs on down-counting, it means the current clock is slower than correct clock, the TRIMVALUE will be increased automatically to raise the clock frequency. Vice versa when it occurs on up-counting, the TRIMVALUE will be reduced automatically to reduce the clock frequency.

- Counter < CKLIM when REF sync pulse is detected.

  The CKOKIF in CTC_STAT register set, and an interrupt generated if CKOKIE bit in CTC_CTL0 register is 1.

If the AUTOTRIM bit in CTC_CTL0 register set, the TRIMVALUE in CTC_CTL0 register is not changed.

- CKLIM ≤ Counter < 3 x CKLIM when REF sync pulse is detected.

  The CKOKIF in CTC_STAT register set, and an interrupt generated if CKOKIE bit in CTC_CTL0 register is 1.

  If the AUTOTRIM bit in CTC_CTL0 register set, the TRIMVALUE in CTC_CTL0 register add 1 when down-counting or sub 1 when up-counting.

- 3 x CKLIM ≤ Counter < 128 x CKLIM when REF sync pulse is detected.

  The CKWARNIF in CTC_STAT register set, and an interrupt generated if CKWARNIE bit in CTC_CTL0 register is 1.

  If the AUTOTRIM bit in CTC_CTL0 register set, the TRIMVALUE in CTC_CTL0 register add 2 when down-counting or sub 2 when up-counting.

- Counter ≥ 128 x CKLIM when down-counting when a REF sync pulse is detected.

  The CKERR in CTC_STAT register set, and an interrupt generated if ERRIE bit in CTC_CTL0 register is 1.

  The TRIMVALUE in CTC_CTL0 register is not changed

- Counter = 128 x CKLIM when up-counting.

  The REFMISS in CTC_STAT register set, and an interrupt generated if ERRIE bit in CTC_CTL0 register is 1.

  The TRIMVALUE in CTC_CTL0 register is not changed.

If adjusting the TRIMVALUE in CTC_CTL0 register over the value of 63, the overflow will be occurred, while adjusting the TRIMVALUE under the value of 0, the underflow will be occurred. The TRIMVALUE is in the range 0 to 63 (the TRIMVALUE is 63 if overflow, the TRIMVALUE is 0 if underflow). Then, the TRIMERR in CTC_STAT register will be set, and an interrupt generated if ERRIE bit in CTC_CTL0 register is 1.

### 7.3.4. Software program guide

The RLVALUE and CKLIM bits in CTC_CTL1 register is critical to evaluate the clock frequency and automatically hardware trim. The value is calculated by the correct clock frequency (IRC48M:48 MHz) and the frequency of REF sync pulse. The ideal case is REF sync pulse occur when the CTC counter is zero, so the RLVALUE is:

$$RLVALUE = (F_{clock} \div F_{REF}) - 1 \quad (7\text{-}1)$$

The CKLIM is set by user according to the clock accuracy. It is recommend to set to the half of the step size, so the CKLIM is:

$$CKLIM=(F_{clock} \div F_{REF}) \times 0.12\% \div 2 \quad (7\text{-}2)$$

The typical step size is 0.12%. Where the $F_{clock}$ is the frequency of correct clock (IRC48M), the $F_{REF}$ is the frequency of reference sync pulse.

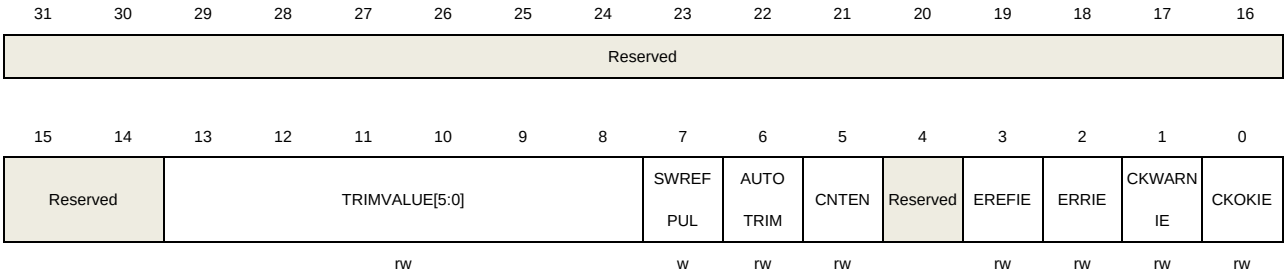
## 7.4. Register definition

CTC base address: 0x4000 8400

### 7.4.1. Control register 0 (CTC_CTL0)

Address offset: 0x00
Reset value: 0x0000 2000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:8 | TRIMVALUE[5:0] | IRC48M trim value<br>When AUTOTRIM in CTC_CTL0 register is 0, these bits are set and cleared by software. This mode used to software calibration.<br>When AUTOTRIM in CTC_CTL0 register is 1, these bits are read only. The value automatically modified by hardware. This mode used to hardware trim.<br>The middle value is 32. When increase 1, the IRC48M clock frequency add around 57KHz. When decrease 1, the IRC48M clock frequency sub around 57KHz. |
| 7 | SWREFPUL | Software reference source sync pulse<br>This bit is set by software, and generates a reference sync pulse to CTC counter. This bit is cleared by hardware automatically and read as 0.<br>0: No effect<br>1: generates a software reference source sync pulse |
| 6 | AUTOTRIM | Hardware automatically trim mode<br>This bit is set and cleared by software. When this bit is set, the hardware automatic trim enabled, the TRIMVALUE bits in CTC_CTL0 register are modified by hardware automatically, until the frequency of IRC48M clock is close to 48MHz.<br>0: Hardware automatic trim disabled<br>1: Hardware automatic trim enabled |
| 5 | CNTEN | CTC counter enable<br>This bit is set and cleared by software. This bit used to enable or disable the CTC trim counter. When this bit is set, the CTC_CTL1 register cannot be modified.<br>0: CTC trim counter disabled |

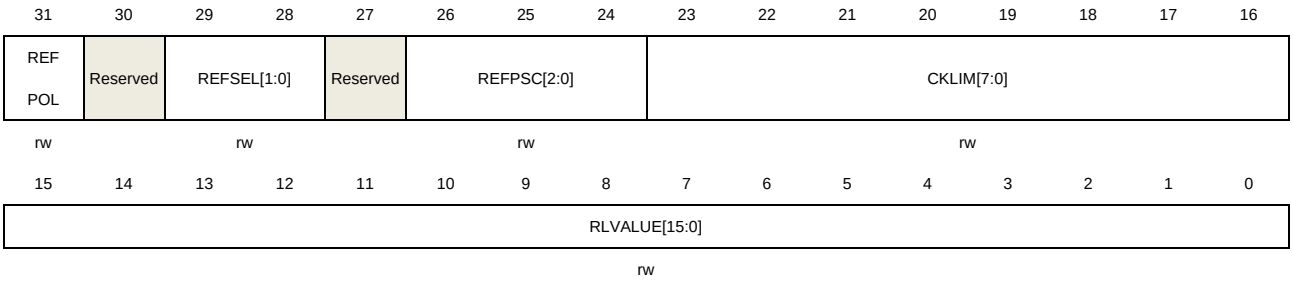
| | | |
|---|---|---|
| | | 1: CTC trim counter enabled. |
| 4 | Reserved | Must be kept at reset value. |
| 3 | EREFIE | EREFIF interrupt enable<br>0: EREFIF interrupt disable<br>1: EREFIF interrupt enable |
| 2 | ERRIE | Error (ERRIF) interrupt enable<br>0: ERRIF interrupt disable<br>1: ERRIF interrupt enable |
| 1 | CKWARNIE | Clock trim warning (CKWARNIF) interrupt enable<br>0: CKWARNIF interrupt disable<br>1: CKWARNIF interrupt enable |
| 0 | CKOKIE | Clock trim OK (CKOKIF) interrupt enable<br>0: CKOKIF interrupt disable<br>1: CKOKIF interrupt enable |

### 7.4.2. Control register 1 (CTC_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

This register has to be accessed by word (32-bit). This register cannot be modified when CNTEN is 1.




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | REFPOL | Reference signal source polarity<br>This bit is set and cleared by software to select reference signal source polarity<br>0: rising edge selected<br>1: falling edge selected |
| 30 | Reserved | Must be kept at reset value. |
| 29:28 | REFSEL[1:0] | Reference signal source selection<br>These bits are set and cleared by software to select reference signal source.<br>00: GPIO(CTC_SYNC) selected<br>01: LXTAL clock selected |

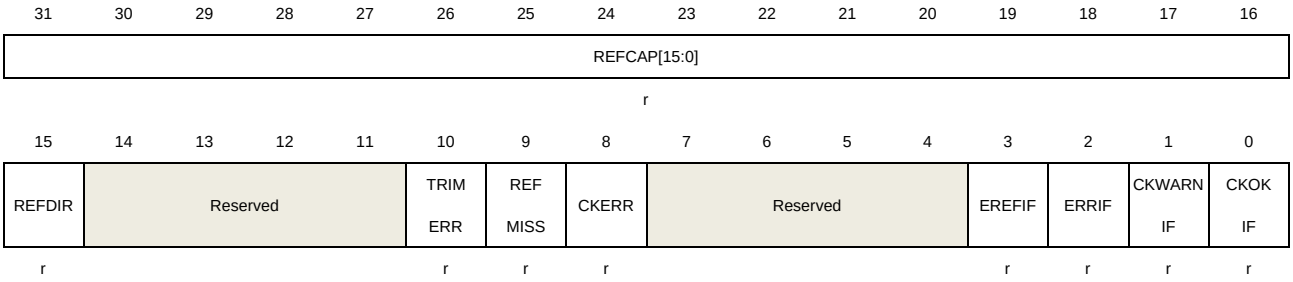
| | | |
|---|---|---|
| | | Other values are reserved. |
| 27 | Reserved | Must be kept at reset value. |
| 26:24 | REFPSC[2:0] | Reference signal source prescaler<br>These bits are set and cleared by software<br>000: Reference signal not divided<br>001: Reference signal divided by 2<br>010: Reference signal divided by 4<br>011: Reference signal divided by 8<br>100: Reference signal divided by 16<br>101: Reference signal divided by 32<br>110: Reference signal divided by 64<br>111: Reference signal divided by 128 |
| 23:16 | CKLIM[7:0] | Clock trim base limit value<br>These bits are set and cleared by software to define the clock trim base limit value. These bits used to frequency evaluation and automatically trim process. Please refer to the ***Frequency evaluation and automatically trim process*** for detail. |
| 15:0 | RLVALUE[15:0] | CTC counter reload value<br>These bits are set and cleared by software to define the CTC counter reload value. These bits reload to CTC trim counter when a reference sync pulse received to start or restart the counter. |

### 7.4.3. Status register (CTC_STAT)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | REFCAP[15:0] | CTC counter capture when reference sync pulse.<br>When a reference sync pulse occurred, the CTC trim counter value is captured to REFCAP bits. |
| 15 | REFDIR | CTC trim counter direction when reference sync pulse<br>When a reference sync pulse occurred during the counter is working, the CTC trim counter direction is captured to REFDIR bit. |

| | | |
|---|---|---|
| | | 0: Up-counting<br>1: Down-counting |
| 14:11 | Reserved | Must be kept at reset value. |
| 10 | TRIMERR | Trim value error bit<br>This bit is set by hardware when the TRIMVALUE in CTC_CTL0 register overflow or underflow. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.<br>0: No trim value error occur<br>1: Trim value error occur |
| 9 | REFMISS | Reference sync pulse miss<br>This bit is set by hardware when the reference sync pulse miss. This is occur when the CTC trim counter reach to 128 x CKLIM during up counting and no reference sync pulse detected. This means the clock is too fast to be trimmed to correct frequency or other error occur. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.<br>0: No Reference sync pulse miss occur<br>1: Reference sync pulse miss occur |
| 8 | CKERR | Clock trim error bit<br>This bit is set by hardware when the clock trim error occur. This is occur when the CTC trim counter greater or equal to 128 x CKLIM during down counting when a reference sync pulse detected. This means the clock is too slow and cannot be trimmed to correct frequency. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.<br>0: No Clock trim error occur<br>1: Clock trim error occur |
| 7:4 | Reserved | Must be kept at reset value. |
| 3 | EREFIF | Expect reference interrupt flag<br>This bit is set by hardware when the CTC counter reach to 0. When the EREFIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to EREFIC bit in CTC_INTC register.<br>0 : No Expect reference occur<br>1: Expect reference occur |
| 2 | ERRIF | Error interrupt flag<br>This bit is set by hardware when an error occurred. If any error of TRIMERR, REFMISS or CKERR occurred, this bit will be set. When the ERRIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.<br>0 : No Error occur<br>1: An error occur |
| 1 | CKWARNIF | Clock trim warning interrupt flag |

This bit is set by hardware when a clock trim warning occurred. If the CTC trim counter greater or equal to 3 x CKLIM and smaller to 128 x CKLIM when a reference sync pulse detected, this bit will be set. This means the clock is too slow or too fast, but can be trim to correct frequency. The TRIMVALUE add 2 or sub 2 when a clock trim warning occurred. When the CKWARNIE in CTC_CTL0 register is set, an interrupt occur. This bit is cleared by writing 1 to CKWARNIC bit in CTC_INTC register.
0 : No Clock trim warning occur
1: Clock trim warning occur

| | | |
|---|---|---|
| 0 | CKOKIF | Clock trim OK interrupt flag<br>This bit is set by hardware when the clock trim is OK. If the CTC trim counter smaller to 3 x CKLIM when a reference sync pulse detected, this bit will be set. This means the clock is OK to use. The TRIMVALUE need not to adjust or adjust one step. When the CKOKIE in CTC_CTL0 register is, an interrupt occur. This bit is cleared by writing 1 to CKOKIC bit in CTC_INTC register.<br>0 : No Clock trim OK occur<br>1: Clock trim OK occur |

### 7.4.4. Interrupt clear register (CTC_INTC)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | EREFIC | ERRIC | CKWARNIC | CKOKIC |
| | | | | | | | | | | | | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | EREFIC | EREFIF interrupt clear bit<br>This bit is written by software and read as 0. Write 1 to clear EREFIF bit in CTC_STAT register. Write 0 is no effect. |
| 2 | ERRIC | ERRIF interrupt clear bit<br>This bit is written by software and read as 0. Write 1 to clear ERRIF, TRIMERR, REFMISS and CKERR bits in CTC_STAT register. Write 0 is no effect. |
| 1 | CKWARNIC | CKWARNIF interrupt clear bit<br>This bit is written by software and read as 0. Write 1 to clear CKWARNIF bit in |

| | | |
|---|---|---|
| | | CTC_STAT register. Write 0 is no effect. |
| 0 | CKOKIC | CKOKIF interrupt clear bit<br>This bit is written by software and read as 0. Write 1 to clear CKOKIF bit in CTC_STAT register. Write 0 is no effect. |

# 8. Interrupt / event controller (EXTI)

## 8.1. Overview

Cortex®-M7 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and power management controls. It's tightly coupled to the processer core. More details about NVIC could refer to the technical reference manual of Cortex®-M7.

EXTI (interrupt / event controller) contains up to 38 independent edge detectors and generates interrupt requests or events to the processer. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

## 8.2. Characteristics

- Cortex®-M7 system exception.
- GD32H737_757_759 support 217 maskable peripheral interrupts (don't contain the 16 interrupts of Cortex®-M7 with FPU); GD32H75E supports 215.
- 4 bits interrupt priority configuration—16 priority levels.
- Efficient interrupt processing.
- Support exception pre-emption and tail-chaining.
- Wake up system from power saving mode.
- Up to 38 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

## 8.3. Interrupts function overview

The Arm® Cortex®-M7 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).

The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. The following tables list all exception types.

**Table 8-1. NVIC exception types in Cortex®-M7**

| Exception type | Vector number | Priority (a) | Vector address | Description |
|---|---|---|---|---|
| - | 0 | - | 0x0000_0000 | Reserved |
| Reset | 1 | -3 | 0x0000_0004 | Reset |
| NMI | 2 | -2 | 0x0000_0008 | Non maskable interrupt. |
| HardFault | 3 | -1 | 0x0000_000C | All class of fault |
| MemManage | 4 | Programmable | 0x0000_0010 | Memory management |
| BusFault | 5 | Programmable | 0x0000_0014 | Prefetch fault, memory access fault |
| UsageFault | 6 | Programmable | 0x0000_0018 | Undefined instruction or illegal state |
| - | 7-10 | - | 0x0000_001C - 0x0000_002B | Reserved |
| SVCall | 11 | Programmable | 0x0000_002C | System service call via SWI instruction |
| Debug Monitor | 12 | Programmable | 0x0000_0030 | Debug monitor |
| - | 13 | - | 0x0000_0034 | Reserved |
| PendSV | 14 | Programmable | 0x0000_0038 | Pendable request for system service |
| SysTick | 15 | Programmable | 0x0000_003C | System tick timer |

The SysTick calibration value is fixed to 1000 and SysTick clock source is from CK_SYS or CK_SYS/8. The SYST_RVR register is configured to provide a 1ms time-base for the system. When SysTick clock source is from CK_SYS (CK_SYS=a MHz), the SYST_RVR register value is set to (a*1000-1). When the SysTick clock source is from CK_SYS/8, the SYST_RVR register value is set to (a/8*1000-1).

**Table 8-2. Interrupt vector table (For GD32H737_757_759)**

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 0** | 16 | WWDGT interrupt | 0x0000_0040 |
| **IRQ 1** | 17 | VAVD/LVD/VOVD through EXTI line detection interrupt | 0x0000_0044 |
| **IRQ 2** | 18 | RTC tamper and timestamp from EXTI Interrupt, LXTAL clock stuck interrupt | 0x0000_0048 |
| **IRQ 3** | 19 | RTC wakeup from EXTI interrupt | 0x0000_004C |
| **IRQ 4** | 20 | FMC global interrupt | 0x0000_0050 |
| **IRQ 5** | 21 | RCU global interrupt | 0x0000_0054 |
| **IRQ 6** | 22 | EXTI line0 interrupt | 0x0000_0058 |
| **IRQ 7** | 23 | EXTI line1 interrupt | 0x0000_005C |
| **IRQ 8** | 24 | EXTI line2 interrupt | 0x0000_0060 |
| **IRQ 9** | 25 | EXTI line3 interrupt | 0x0000_0064 |
| **IRQ 10** | 26 | EXTI line4 interrupt | 0x0000_0068 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 11** | 27 | DMA0 channel0 global interrupt | 0x0000_006C |
| **IRQ 12** | 28 | DMA0 channel1 global interrupt | 0x0000_0070 |
| **IRQ 13** | 29 | DMA0 channel2 global interrupt | 0x0000_0074 |
| **IRQ 14** | 30 | DMA0 channel3 global interrupt | 0x0000_0078 |
| **IRQ 15** | 31 | DMA0 channel4 global interrupt | 0x0000_007C |
| **IRQ 16** | 32 | DMA0 channel5 global interrupt | 0x0000_0080 |
| **IRQ 17** | 33 | DMA0 channel6 global interrupt | 0x0000_0084 |
| **IRQ 18** | 34 | ADC0 and ADC1 global interrupt | 0x0000_0088 |
| **IRQ 19-22** | 35-38 | Reserved | 0x0000_008C-0x0000_0098 |
| **IRQ 23** | 39 | EXTI line5-9 interrupt | 0x0000_009C |
| **IRQ 24** | 40 | TIMER0 break interrupt | 0x0000_00A0 |
| **IRQ 25** | 41 | TIMER0 update interrupt | 0x0000_00A4 |
| **IRQ 26** | 42 | TIMER0 trigger and commutation interrupt | 0x0000_00A8 |
| **IRQ 27** | 43 | TIMER0 capture compare interrupt | 0x0000_00AC |
| **IRQ 28** | 44 | TIMER1 global interrupt | 0x0000_00B0 |
| **IRQ 29** | 45 | TIMER2 global interrupt | 0x0000_00B4 |
| **IRQ 30** | 46 | TIMER3 global interrupt | 0x0000_00B8 |
| **IRQ 31** | 47 | I2C0 event interrupt | 0x0000_00BC |
| **IRQ 32** | 48 | I2C0 error interrupt | 0x0000_00C0 |
| **IRQ 33** | 49 | I2C1 event interrupt | 0x0000_00C4 |
| **IRQ 34** | 50 | I2C1 error interrupt | 0x0000_00C8 |
| **IRQ 35** | 51 | SPI0 global interrupt | 0x0000_00CC |
| **IRQ 36** | 52 | SPI1 global interrupt | 0x0000_00D0 |
| **IRQ 37** | 53 | USART0 global and wakeup interrupt | 0x0000_00D4 |
| **IRQ 38** | 54 | USART1 global and wakeup interrupt | 0x0000_00D8 |
| **IRQ 39** | 55 | USART2 global and wakeup interrupt | 0x0000_00DC |
| **IRQ 40** | 56 | EXTI line10-15 interrupt | 0x0000_00E0 |
| **IRQ 41** | 57 | RTC alarm from EXTI interrupt | 0x0000_00E4 |
| **IRQ 42** | 58 | Reserved | 0x0000_00E8 |
| **IRQ 43** | 59 | TIMER7 break interrupt | 0x0000_00EC |
| **IRQ 44** | 60 | TIMER7 update interrupt | 0x0000_00F0 |
| **IRQ 45** | 61 | TIMER7 trigger and commutation interrupt | 0x0000_00F4 |
| **IRQ 46** | 62 | TIMER7 capture compare interrupt | 0x0000_00F8 |
| **IRQ 47** | 63 | DMA0 channel7 global interrupt | 0x0000_00FC |
| **IRQ 48** | 64 | EXMC global interrupt | 0x0000_0100 |
| **IRQ 49** | 65 | SDIO0 global interrupt | 0x0000_0104 |
| **IRQ 50** | 66 | TIMER4 global interrupt | 0x0000_0108 |
| **IRQ 51** | 67 | SPI2 global interrupt | 0x0000_010C |
| **IRQ 52** | 68 | UART3 global interrupt | 0x0000_0110 |
| **IRQ 53** | 69 | UART4 global interrupt | 0x0000_0114 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 54** | 70 | TIMER5 global interrupt and DAC underrun error interrupt | 0x0000_0118 |
| **IRQ 55** | 71 | TIMER6 global interrupt | 0x0000_011C |
| **IRQ 56** | 72 | DMA1 channel 0 global interrupt | 0x0000_0120 |
| **IRQ 57** | 73 | DMA1 channel 1 global interrupt | 0x0000_0124 |
| **IRQ 58** | 74 | DMA1 channel 2 global interrupt | 0x0000_0128 |
| **IRQ 59** | 75 | DMA1 channel 3 global interrupt | 0x0000_012C |
| **IRQ 60** | 76 | DMA1 channel 4 global interrupt | 0x0000_0130 |
| **IRQ 61** | 77 | Ethernet0 global interrupt | 0x0000_0134 |
| **IRQ 62** | 78 | Ethernet0 wakeup from EXTI interrupt | 0x0000_0138 |
| **IRQ 63-67** | 79-83 | Reserved | 0x0000_013C-0x0000_014C |
| **IRQ 68** | 84 | DMA1 channel 5 global interrupt | 0x0000_0150 |
| **IRQ 69** | 85 | DMA1 channel 6 global interrupt | 0x0000_0154 |
| **IRQ 70** | 86 | DMA1 channel 7 global interrupt | 0x0000_0158 |
| **IRQ 71** | 87 | USART5 global and wakeup interrupt | 0x0000_015C |
| **IRQ 72** | 88 | I2C2 event interrupt | 0x0000_0160 |
| **IRQ 73** | 89 | I2C2 error interrupt | 0x0000_0164 |
| **IRQ 74** | 90 | USBHS0 endpoint 1 out interrupt | 0x0000_0168 |
| **IRQ 75** | 91 | USBHS0 endpoint 1 in interrupt | 0x0000_016C |
| **IRQ 76** | 92 | USBHS0 wakeup from EXTI interrupt | 0x0000_0170 |
| **IRQ 77** | 93 | USBHS0 global interrupt | 0x0000_0174 |
| **IRQ 78** | 94 | DCI global interrupt | 0x0000_0178 |
| **IRQ 79** | 95 | CAU global interrupt | 0x0000_017C |
| **IRQ 80** | 96 | HAU and TRNG global interrupt | 0x0000_0180 |
| **IRQ 81** | 97 | FPU global interrupt | 0x0000_0184 |
| **IRQ 82** | 98 | UART6 global interrupt | 0x0000_0188 |
| **IRQ 83** | 99 | UART7 global interrupt | 0x0000_018C |
| **IRQ 84** | 100 | SPI3 global interrupt | 0x0000_0190 |
| **IRQ 85** | 101 | SPI4 global interrupt | 0x0000_0194 |
| **IRQ 86** | 102 | SPI5 global interrupt | 0x0000_0198 |
| **IRQ 87** | 103 | SAI0 global interrupt | 0x0000_019C |
| **IRQ 88** | 104 | TLI global interrupt | 0x0000_01A0 |
| **IRQ 89** | 105 | TLI error interrupt | 0x0000_01A4 |
| **IRQ 90** | 106 | IPA global interrupt | 0x0000_01A8 |
| **IRQ 91** | 107 | SAI1 global interrupt | 0x0000_01AC |
| **IRQ 92** | 108 | OSPI0 global interrupt | 0x0000_01B0 |
| **IRQ 93-94** | 109-110 | Reserved | 0x0000_01B4-0x0000_01B8 |
| **IRQ 95** | 111 | I2C3 event interrupt | 0x0000_01BC |
| **IRQ 96** | 112 | I2C3 error interrupt | 0x0000_01C0 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 97** | 113 | RSPDIF global interrupt | 0x0000_01C4 |
| **IRQ 98-101** | 114-117 | Reserved | 0x0000_01C8-0x0000_01D4 |
| **IRQ 102** | 118 | DMAMUX overrun interrupt | 0x0000_01D8 |
| **IRQ 103-109** | 119-125 | Reserved | 0x0000_01DC-0x0000_01F4 |
| **IRQ 110** | 126 | HPDF global interrupt 0 | 0x0000_01F8 |
| **IRQ 111** | 127 | HPDF global interrupt 1 | 0x0000_01FC |
| **IRQ 112** | 128 | HPDF global interrupt 2 | 0x0000_0200 |
| **IRQ 113** | 129 | HPDF global interrupt 3 | 0x0000_0204 |
| **IRQ 114** | 130 | SAI2 global interrupt | 0x0000_0208 |
| **IRQ 115** | 131 | Reserved | 0x0000_020C |
| **IRQ 116** | 132 | TIMER14 global interrupt | 0x0000_0210 |
| **IRQ 117** | 133 | TIMER15 global interrupt | 0x0000_0214 |
| **IRQ 118** | 134 | TIMER16 global interrupt | 0x0000_0218 |
| **IRQ 119** | 135 | Reserved | 0x0000_021C |
| **IRQ 120** | 136 | MDIO global interrupt | 0x0000_0220 |
| **IRQ 121** | 137 | Reserved | 0x0000_0224 |
| **IRQ 122** | 138 | MDMA global interrupt | 0x0000_0228 |
| **IRQ 123** | 139 | Reserved | 0x0000_022C |
| **IRQ 124** | 140 | SDIO1 global interrupt | 0x0000_0230 |
| **IRQ 125** | 141 | HWSEM global interrupt | 0x0000_0234 |
| **IRQ 126** | 142 | Reserved | 0x0000_0238 |
| **IRQ 127** | 143 | ADC2 global interrupt | 0x0000_023C |
| **IRQ 128-136** | 144-152 | Reserved | 0x0000_0240-0x0000_0260 |
| **IRQ 137** | 153 | CMP0 and CMP1 global interrupt, CMP0 and CMP1 through EXTI line detection interrupt | 0x0000_0264 |
| **IRQ 138-143** | 154-159 | Reserved | 0x0000_0268-0x0000_027C |
| **IRQ 144** | 160 | CTC interrupt | 0x0000_0280 |
| **IRQ 145** | 161 | RAMECCMU global interrupt | 0x0000_0284 |
| **IRQ 146-149** | 162-165 | Reserved | 0x0000_0288-0x0000_0294 |
| **IRQ 150** | 166 | OSPI1 global interrupt | 0x0000_0298 |
| **IRQ 151** | 167 | RTDEC0 global interrupt | 0x0000_029C |
| **IRQ 152** | 168 | RTDEC1 global interrupt | 0x0000_02A0 |
| **IRQ 153** | 169 | FAC global interrupt | 0x0000_02A4 |
| **IRQ 154** | 170 | TMU global interrupt | 0x0000_02A8 |
| **IRQ 155-160** | 171-176 | Reserved | 0x0000_02AC-0x0000_02C0 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 161** | 177 | TIMER22 global interrupt | 0x0000_02C4 |
| **IRQ 162** | 178 | TIMER23 global interrupt | 0x0000_02C8 |
| **IRQ 163** | 179 | TIMER30 global interrupt | 0x0000_02CC |
| **IRQ 164** | 180 | TIMER31 global interrupt | 0x0000_02D0 |
| **IRQ 165** | 181 | TIMER40 global interrupt | 0x0000_02D4 |
| **IRQ 166** | 182 | TIMER41 global interrupt | 0x0000_02D8 |
| **IRQ 167** | 183 | TIMER42 global interrupt | 0x0000_02DC |
| **IRQ 168** | 184 | TIMER43 global interrupt | 0x0000_02E0 |
| **IRQ 169** | 185 | TIMER44 global interrupt | 0x0000_02E4 |
| **IRQ 170** | 186 | TIMER50 global interrupt | 0x0000_02E8 |
| **IRQ 171** | 187 | TIMER51 global interrupt | 0x0000_02EC |
| **IRQ 172** | 188 | USBHS1 endpoint 1 out interrupt | 0x0000_02F0 |
| **IRQ 173** | 189 | USBHS1 endpoint 1 in interrupt | 0x0000_02F4 |
| **IRQ 174** | 190 | USBHS1 wakeup from EXTI interrupt | 0x0000_02F8 |
| **IRQ 175** | 191 | USBHS1 global interrupt | 0x0000_02FC |
| **IRQ 176** | 192 | Ethernet1 global interrupt | 0x0000_0300 |
| **IRQ 177** | 193 | Ethernet1 wakeup from EXTI interrupt | 0x0000_0304 |
| **IRQ 178** | 194 | Reserved | 0x0000_0308 |
| **IRQ 179** | 195 | CAN0 wakeup through EXTI line detection interrupt | 0x0000_030C |
| **IRQ 180** | 196 | CAN0 interrupt for message buffer | 0x0000_0310 |
| **IRQ 181** | 197 | CAN0 interrupt for Bus off / Bus off done | 0x0000_0314 |
| **IRQ 182** | 198 | CAN0 interrupt for Error | 0x0000_0318 |
| **IRQ 183** | 199 | CAN0 interrupt for Error in fast transmission | 0x0000_031C |
| **IRQ 184** | 200 | CAN0 interrupt for Transmit warning | 0x0000_0320 |
| **IRQ 185** | 201 | CAN0 interrupt for Receive warning | 0x0000_0324 |
| **IRQ 186** | 202 | CAN1 wakeup through EXTI line detection interrupt | 0x0000_0328 |
| **IRQ 187** | 203 | CAN1 interrupt for message buffer | 0x0000_032C |
| **IRQ 188** | 204 | CAN1 interrupt for Bus off / Bus off done | 0x0000_0330 |
| **IRQ 189** | 205 | CAN1 interrupt for Error | 0x0000_0334 |
| **IRQ 190** | 206 | CAN1 interrupt for Error in fast transmission | 0x0000_0338 |
| **IRQ 191** | 207 | CAN1 interrupt for Transmit warning | 0x0000_033C |
| **IRQ 192** | 208 | CAN1 interrupt for Receive warning | 0x0000_0340 |
| **IRQ 193** | 209 | CAN2 wakeup through EXTI line detection interrupt | 0x0000_0344 |
| **IRQ 194** | 210 | CAN2 interrupt for message buffer | 0x0000_0348 |
| **IRQ 195** | 211 | CAN2 interrupt for Bus off / Bus off done | 0x0000_034C |
| **IRQ 196** | 212 | CAN2 interrupt for Error | 0x0000_0350 |
| **IRQ 197** | 213 | CAN2 interrupt for Error in fast transmission | 0x0000_0354 |
| **IRQ 198** | 214 | CAN2 interrupt for Transmit warning | 0x0000_0358 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 199** | 215 | CAN2 interrupt for Receive warning | 0x0000_035C |
| **IRQ 200** | 216 | EFUSE global interrupt | 0x0000_0360 |
| **IRQ 201** | 217 | I2C0 wakeup through EXTI line detection interrupt | 0x0000_0364 |
| **IRQ 202** | 218 | I2C1 wakeup through EXTI line detection interrupt | 0x0000_0368 |
| **IRQ 203** | 219 | I2C2 wakeup through EXTI line detection interrupt | 0x0000_036C |
| **IRQ 204** | 220 | I2C3 wakeup through EXTI line detection interrupt | 0x0000_0370 |
| **IRQ 205** | 221 | LPDTS interrupt | 0x0000_0374 |
| **IRQ 206** | 222 | LPDTS wakeup through EXTI line detection interrupt | 0x0000_0378 |
| **IRQ 207** | 223 | TIMER0 DEC interrupt | 0x0000_037C |
| **IRQ 208** | 224 | TIMER7 DEC interrupt | 0x0000_0380 |
| **IRQ 209** | 225 | TIMER1 DEC interrupt | 0x0000_0384 |
| **IRQ 210** | 226 | TIMER2 DEC interrupt | 0x0000_0388 |
| **IRQ 211** | 227 | TIMER3 DEC interrupt | 0x0000_038C |
| **IRQ 212** | 228 | TIMER4 DEC interrupt | 0x0000_0390 |
| **IRQ 213** | 229 | TIMER22 DEC interrupt | 0x0000_0394 |
| **IRQ 214** | 230 | TIMER23 DEC interrupt | 0x0000_0398 |
| **IRQ 215** | 231 | TIMER30 DEC interrupt | 0x0000_039C |
| **IRQ 216** | 232 | TIMER31 DEC interrupt | 0x0000_03A0 |

**Table 8-3. Interrupt vector table (For GD32H75E)**

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 0** | 16 | WWDGT interrupt | 0x0000_0040 |
| **IRQ 1** | 17 | VAVD/LVD/VOVD through EXTI line detection interrupt | 0x0000_0044 |
| **IRQ 2** | 18 | RTC tamper and timestamp from EXTI Interrupt, LXTAL clock stuck interrupt | 0x0000_0048 |
| **IRQ 3** | 19 | RTC wakeup from EXTI interrupt | 0x0000_004C |
| **IRQ 4** | 20 | FMC global interrupt | 0x0000_0050 |
| **IRQ 5** | 21 | RCU global interrupt | 0x0000_0054 |
| **IRQ 6** | 22 | EXTI line0 interrupt | 0x0000_0058 |
| **IRQ 7** | 23 | EXTI line1 interrupt | 0x0000_005C |
| **IRQ 8** | 24 | EXTI line2 interrupt | 0x0000_0060 |
| **IRQ 9** | 25 | EXTI line3 interrupt | 0x0000_0064 |
| **IRQ 10** | 26 | EXTI line4 interrupt | 0x0000_0068 |
| **IRQ 11** | 27 | DMA0 channel0 global interrupt | 0x0000_006C |
| **IRQ 12** | 28 | DMA0 channel1 global interrupt | 0x0000_0070 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 13** | 29 | DMA0 channel2 global interrupt | 0x0000_0074 |
| **IRQ 14** | 30 | DMA0 channel3 global interrupt | 0x0000_0078 |
| **IRQ 15** | 31 | DMA0 channel4 global interrupt | 0x0000_007C |
| **IRQ 16** | 32 | DMA0 channel5 global interrupt | 0x0000_0080 |
| **IRQ 17** | 33 | DMA0 channel6 global interrupt | 0x0000_0084 |
| **IRQ 18** | 34 | ADC0 and ADC1 global interrupt | 0x0000_0088 |
| **IRQ 19-22** | 35-38 | Reserved | 0x0000_008C-0x0000_0098 |
| **IRQ 23** | 39 | EXTI line5-9 interrupt | 0x0000_009C |
| **IRQ 24** | 40 | TIMER0 break interrupt | 0x0000_00A0 |
| **IRQ 25** | 41 | TIMER0 update interrupt | 0x0000_00A4 |
| **IRQ 26** | 42 | TIMER0 trigger and commutation interrupt | 0x0000_00A8 |
| **IRQ 27** | 43 | TIMER0 capture compare interrupt | 0x0000_00AC |
| **IRQ 28** | 44 | TIMER1 global interrupt | 0x0000_00B0 |
| **IRQ 29** | 45 | TIMER2 global interrupt | 0x0000_00B4 |
| **IRQ 30** | 46 | TIMER3 global interrupt | 0x0000_00B8 |
| **IRQ 31** | 47 | I2C0 event interrupt | 0x0000_00BC |
| **IRQ 32** | 48 | I2C0 error interrupt | 0x0000_00C0 |
| **IRQ 33** | 49 | I2C1 event interrupt | 0x0000_00C4 |
| **IRQ 34** | 50 | I2C1 error interrupt | 0x0000_00C8 |
| **IRQ 35** | 51 | SPI0 global interrupt | 0x0000_00CC |
| **IRQ 36** | 52 | SPI1 global interrupt | 0x0000_00D0 |
| **IRQ 37** | 53 | USART0 global and wakeup interrupt | 0x0000_00D4 |
| **IRQ 38** | 54 | USART1 global and wakeup interrupt | 0x0000_00D8 |
| **IRQ 39** | 55 | USART2 global and wakeup interrupt | 0x0000_00DC |
| **IRQ 40** | 56 | EXTI line10-15 interrupt | 0x0000_00E0 |
| **IRQ 41** | 57 | RTC alarm from EXTI interrupt | 0x0000_00E4 |
| **IRQ 42** | 58 | Reserved | 0x0000_00E8 |
| **IRQ 43** | 59 | TIMER7 break interrupt | 0x0000_00EC |
| **IRQ 44** | 60 | TIMER7 update interrupt | 0x0000_00F0 |
| **IRQ 45** | 61 | TIMER7 trigger and commutation interrupt | 0x0000_00F4 |
| **IRQ 46** | 62 | TIMER7 capture compare interrupt | 0x0000_00F8 |
| **IRQ 47** | 63 | DMA0 channel7 global interrupt | 0x0000_00FC |
| **IRQ 48** | 64 | EXMC global interrupt | 0x0000_0100 |
| **IRQ 49** | 65 | Reserved | 0x0000_0104 |
| **IRQ 50** | 66 | TIMER4 global interrupt | 0x0000_0108 |
| **IRQ 51** | 67 | SPI2 global interrupt | 0x0000_010C |
| **IRQ 52** | 68 | UART3 global interrupt | 0x0000_0110 |
| **IRQ 53** | 69 | UART4 global interrupt | 0x0000_0114 |
| **IRQ 54** | 70 | TIMER5 global interrupt and DAC underrun error interrupt | 0x0000_0118 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 55** | 71 | TIMER6 global interrupt | 0x0000_011C |
| **IRQ 56** | 72 | DMA1 channel 0 global interrupt | 0x0000_0120 |
| **IRQ 57** | 73 | DMA1 channel 1 global interrupt | 0x0000_0124 |
| **IRQ 58** | 74 | DMA1 channel 2 global interrupt | 0x0000_0128 |
| **IRQ 59** | 75 | DMA1 channel 3 global interrupt | 0x0000_012C |
| **IRQ 60** | 76 | DMA1 channel 4 global interrupt | 0x0000_0130 |
| **IRQ 61-67** | 77-83 | Reserved | 0x0000_0134-0x0000_014C |
| **IRQ 68** | 84 | DMA1 channel 5 global interrupt | 0x0000_0150 |
| **IRQ 69** | 85 | DMA1 channel 6 global interrupt | 0x0000_0154 |
| **IRQ 70** | 86 | DMA1 channel 7 global interrupt | 0x0000_0158 |
| **IRQ 71** | 87 | USART5 global and wakeup interrupt | 0x0000_015C |
| **IRQ 72** | 88 | I2C2 event interrupt | 0x0000_0160 |
| **IRQ 73** | 89 | I2C2 error interrupt | 0x0000_0164 |
| **IRQ 74** | 90 | USBHS0 endpoint 1 out interrupt | 0x0000_0168 |
| **IRQ 75** | 91 | USBHS0 endpoint 1 in interrupt | 0x0000_016C |
| **IRQ 76** | 92 | USBHS0 wakeup from EXTI interrupt | 0x0000_0170 |
| **IRQ 77** | 93 | USBHS0 global interrupt | 0x0000_0174 |
| **IRQ 78-79** | 94-95 | Reserved | 0x0000_0178-0x0000_017C |
| **IRQ 80** | 96 | TRNG global interrupt | 0x0000_0180 |
| **IRQ 81** | 97 | FPU global interrupt | 0x0000_0184 |
| **IRQ 82** | 98 | UART6 global interrupt | 0x0000_0188 |
| **IRQ 83** | 99 | UART7 global interrupt | 0x0000_018C |
| **IRQ 84** | 100 | SPI3 global interrupt | 0x0000_0190 |
| **IRQ 85** | 101 | SPI4 global interrupt | 0x0000_0194 |
| **IRQ 86** | 102 | SPI5 global interrupt | 0x0000_0198 |
| **IRQ 87-91** | 103-107 | Reserved | 0x0000_019C-0x0000_01AC |
| **IRQ 92** | 108 | OSPI0 global interrupt | 0x0000_01B0 |
| **IRQ 93-94** | 109-110 | Reserved | 0x0000_01B4-0x0000_01B8 |
| **IRQ 95** | 111 | I2C3 event interrupt | 0x0000_01BC |
| **IRQ 96** | 112 | I2C3 error interrupt | 0x0000_01C0 |
| **IRQ 97-101** | 113-117 | Reserved | 0x0000_01C4-0x0000_01D4 |
| **IRQ 102** | 118 | DMAMUX overrun interrupt | 0x0000_01D8 |
| **IRQ 103-109** | 119-125 | Reserved | 0x0000_01DC-0x0000_01F4 |
| **IRQ 110** | 126 | HPDF global interrupt 0 | 0x0000_01F8 |
| **IRQ 111** | 127 | HPDF global interrupt 1 | 0x0000_01FC |

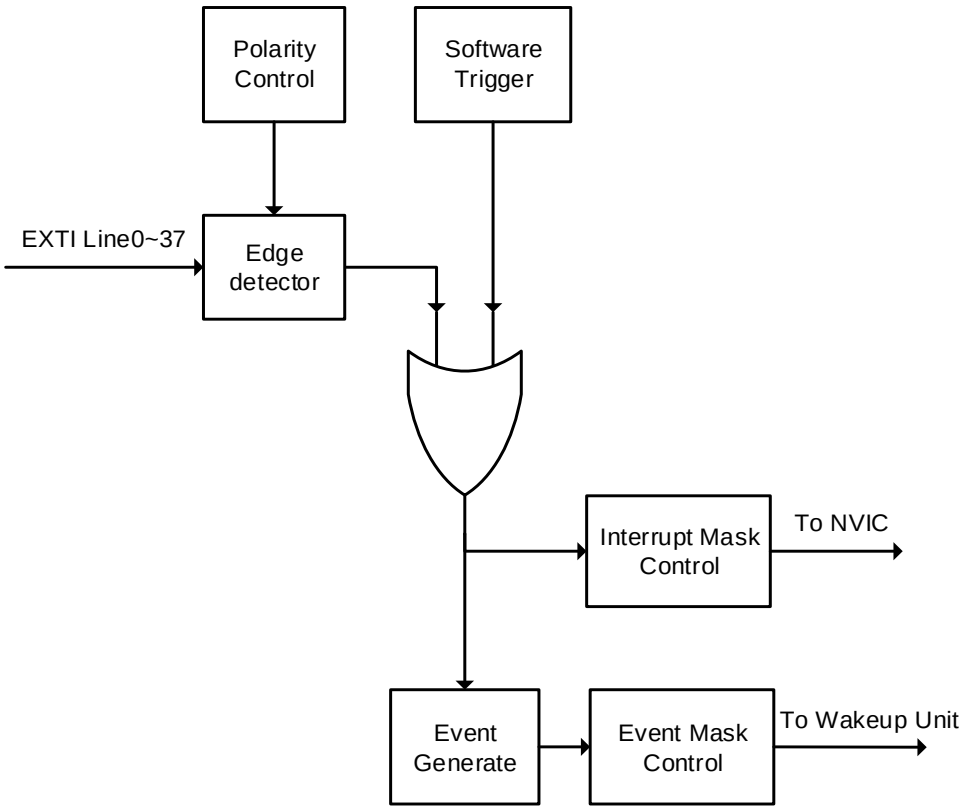
| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 112** | 128 | HPDF global interrupt 2 | 0x0000_0200 |
| **IRQ 113** | 129 | HPDF global interrupt 3 | 0x0000_0204 |
| **IRQ 114-115** | 130-131 | Reserved | 0x0000_0208-0x0000_020C |
| **IRQ 116** | 132 | TIMER14 global interrupt | 0x0000_0210 |
| **IRQ 117** | 133 | TIMER15 global interrupt | 0x0000_0214 |
| **IRQ 118** | 134 | TIMER16 global interrupt | 0x0000_0218 |
| **IRQ 119-121** | 135-137 | Reserved | 0x0000_021C-0x0000_0224 |
| **IRQ 122** | 138 | MDMA global interrupt | 0x0000_0228 |
| **IRQ 123-126** | 139-142 | Reserved | 0x0000_022C-0x0000_0238 |
| **IRQ 127** | 143 | ADC2 global interrupt | 0x0000_023C |
| **IRQ 128-136** | 144-152 | Reserved | 0x0000_0240-0x0000_0260 |
| **IRQ 137** | 153 | CMP0 and CMP1 global interrupt, CMP0 and CMP1 through EXTI line detection interrupt | 0x0000_0264 |
| **IRQ 138-143** | 154-159 | Reserved | 0x0000_0268-0x0000_027C |
| **IRQ 144** | 160 | CTC interrupt | 0x0000_0280 |
| **IRQ 145** | 161 | RAMECCMU global interrupt | 0x0000_0284 |
| **IRQ 146-149** | 162-165 | Reserved | 0x0000_0288-0x0000_0294 |
| **IRQ 150** | 166 | OSPI1 global interrupt | 0x0000_0298 |
| **IRQ 151-152** | 167-168 | Reserved | 0x0000_029C-0x0000_02A0 |
| **IRQ 153** | 169 | FAC global interrupt | 0x0000_02A4 |
| **IRQ 154** | 170 | TMU global interrupt | 0x0000_02A8 |
| **IRQ 155-160** | 171-176 | Reserved | 0x0000_02AC-0x0000_02C0 |
| **IRQ 161** | 177 | TIMER22 global interrupt | 0x0000_02C4 |
| **IRQ 162** | 178 | TIMER23 global interrupt | 0x0000_02C8 |
| **IRQ 163-164** | 179-180 | Reserved | 0x0000_02CC-0x0000_02D0 |
| **IRQ 165** | 181 | TIMER40 global interrupt | 0x0000_02D4 |
| **IRQ 166** | 182 | TIMER41 global interrupt | 0x0000_02D8 |
| **IRQ 167** | 183 | TIMER42 global interrupt | 0x0000_02DC |
| **IRQ 168** | 184 | TIMER43 global interrupt | 0x0000_02E0 |
| **IRQ 169** | 185 | TIMER44 global interrupt | 0x0000_02E4 |
| **IRQ 170** | 186 | TIMER50 global interrupt | 0x0000_02E8 |
| **IRQ 171** | 187 | TIMER51 global interrupt | 0x0000_02EC |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 172** | 188 | USBHS1 endpoint 1 out interrupt | 0x0000_02F0 |
| **IRQ 173** | 189 | USBHS1 endpoint 1 in interrupt | 0x0000_02F4 |
| **IRQ 174** | 190 | USBHS1 wakeup from EXTI interrupt | 0x0000_02F8 |
| **IRQ 175** | 191 | USBHS1 global interrupt | 0x0000_02FC |
| **IRQ 176-178** | 192-194 | Reserved | 0x0000_0300-0x0000_0308 |
| **IRQ 179** | 195 | CAN0 wakeup through EXTI line detection interrupt | 0x0000_030C |
| **IRQ 180** | 196 | CAN0 interrupt for message buffer | 0x0000_0310 |
| **IRQ 181** | 197 | CAN0 interrupt for Bus off / Bus off done | 0x0000_0314 |
| **IRQ 182** | 198 | CAN0 interrupt for Error | 0x0000_0318 |
| **IRQ 183** | 199 | CAN0 interrupt for Error in fast transmission | 0x0000_031C |
| **IRQ 184** | 200 | CAN0 interrupt for Transmit warning | 0x0000_0320 |
| **IRQ 185** | 201 | CAN0 interrupt for Receive warning | 0x0000_0324 |
| **IRQ 186** | 202 | CAN1 wakeup through EXTI line detection interrupt | 0x0000_0328 |
| **IRQ 187** | 203 | CAN1 interrupt for message buffer | 0x0000_032C |
| **IRQ 188** | 204 | CAN1 interrupt for Bus off / Bus off done | 0x0000_0330 |
| **IRQ 189** | 205 | CAN1 interrupt for Error | 0x0000_0334 |
| **IRQ 190** | 206 | CAN1 interrupt for Error in fast transmission | 0x0000_0338 |
| **IRQ 191** | 207 | CAN1 interrupt for Transmit warning | 0x0000_033C |
| **IRQ 192** | 208 | CAN1 interrupt for Receive warning | 0x0000_0340 |
| **IRQ 193** | 209 | CAN2 wakeup through EXTI line detection interrupt | 0x0000_0344 |
| **IRQ 194** | 210 | CAN2 interrupt for message buffer | 0x0000_0348 |
| **IRQ 195** | 211 | CAN2 interrupt for Bus off / Bus off done | 0x0000_034C |
| **IRQ 196** | 212 | CAN2 interrupt for Error | 0x0000_0350 |
| **IRQ 197** | 213 | CAN2 interrupt for Error in fast transmission | 0x0000_0354 |
| **IRQ 198** | 214 | CAN2 interrupt for Transmit warning | 0x0000_0358 |
| **IRQ 199** | 215 | CAN2 interrupt for Receive warning | 0x0000_035C |
| **IRQ 200** | 216 | EFUSE global interrupt | 0x0000_0360 |
| **IRQ 201** | 217 | I2C0 wakeup through EXTI line detection interrupt | 0x0000_0364 |
| **IRQ 202** | 218 | I2C1 wakeup through EXTI line detection interrupt | 0x0000_0368 |
| **IRQ 203** | 219 | I2C2 wakeup through EXTI line detection interrupt | 0x0000_036C |
| **IRQ 204** | 220 | I2C3 wakeup through EXTI line detection interrupt | 0x0000_0370 |
| **IRQ 205** | 221 | LPDTS interrupt | 0x0000_0374 |

| Interrupt number | Vector number | Peripheral interrupt description | Vector address |
|---|---|---|---|
| **IRQ 206** | 222 | LPDTS wakeup through EXTI line detection interrupt | 0x0000_0378 |
| **IRQ 207** | 223 | TIMER0 DEC interrupt | 0x0000_037C |
| **IRQ 208** | 224 | TIMER7 DEC interrupt | 0x0000_0380 |
| **IRQ 209** | 225 | TIMER1 DEC interrupt | 0x0000_0384 |
| **IRQ 210** | 226 | TIMER2 DEC interrupt | 0x0000_0388 |
| **IRQ 211** | 227 | TIMER3 DEC interrupt | 0x0000_038C |
| **IRQ 212** | 228 | TIMER4 DEC interrupt | 0x0000_0390 |
| **IRQ 213** | 229 | TIMER22 DEC interrupt | 0x0000_0394 |
| **IRQ 214** | 230 | TIMER23 DEC interrupt | 0x0000_0398 |

## 8.4. External interrupt and event (EXTI) block diagram

**Figure 8-1. Block diagram of EXTI**




## 8.5. External interrupt and event function overview

The EXTI contains up to 38 independent edge detectors and generates interrupts request or event to the processer. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

The EXTI trigger source includes 16 external lines from GPIO pins and 22 lines from internal modules which refers to ***Table 8-4. EXTI source (For GD32H737_757_759)*** and ***Table 8-5. EXTI source (For GD32H75E)***. All GPIO pins can be selected as an EXTI trigger source by configuring SYSCFG_EXTISSx registers in SYSCFG module (please refer to ***System configuration registers*** for detail).

EXTI can provide not only interrupts but also event signals to the processor. The Cortex®-M7 processor fully implements the Wait For Interrupt (WFI), Wait For Event (WFE) and the Send Event (SEV) instructions. The Wake-up Interrupt Controller (WIC) enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts and events. EXTI can be used to wake up processor and the whole system when some expected event occurs, such as a special GPIO pin toggling or RTC alarm.

### Hardware trigger

Hardware trigger may be used to detect the voltage change of external or internal signals. The software should follow these steps to use this function:

1. Configure EXTI sources in SYSCFG module based on application requirement.
2. Configure EXTI_RTEN and EXTI_FTEN to enable the rising or falling detection on related pins. (Software may set both RTENx and FTENx for a pin at the same time to detect both rising and falling changes on this pin).
3. Enable interrupts or events by setting related EXTI_INTEN or EXTI_EVEN bits.
4. EXTI starts to detect changes on the configured pins. The related interrupt or event will be triggered when desired change is detected on these pins. If the interrupt is triggered, the related PDx is set; if the event is triggered, the related PDx is not set. The software should response to the interrupts or events and clear these PDx bits.

### Software trigger

Software may also trigger EXTI interrupts or events following these steps:

1. Enable interrupts or events by setting related EXTI_INTEN or EXTI_EVEN bits.
2. Set SWIEVx bits in EXTI_SWIEV register, the related interrupt or event will be triggered immediately. If the interrupt is triggered, the related PDx is set; if the event is triggered, the related PDx is not set. Software should response to these interrupts, and clear related PDx bits.

**Table 8-4. EXTI source (For GD32H737_757_759)**

| EXTI line number | Source |
|---|---|
| 0 | PA0 / PB0 / PC0 / PD0 / PE0 / PF0 / PG0 / PH0 / PK0 |
| 1 | PA1 / PB1 / PC1 / PD1 / PE1 / PF1 / PG1 / PH1 / PK1 |
| 2 | PA2 / PB2 / PC2 / PD2 / PE2 / PF2 / PG2 / PH2 / PK2 |
| 3 | PA3 / PB3 / PC3 / PD3 / PE3 / PF3 / PG3 / PH3 |
| 4 | PA4 / PB4 / PC4 / PD4 / PE4 / PF4 / PG4 / PH4 |
| 5 | PA5 / PB5 / PC5 / PD5 / PE5 / PF5 / PG5 / PH5 |
| 6 | PA6 / PB6 / PC6 / PD6 / PE6 / PF6 / PG6 / PH6 |
| 7 | PA7 / PB7 / PC7 / PD7 / PE7 / PF7 / PG7 / PH7 |
| 8 | PA8 / PB8 / PC8 / PD8 / PE8 / PF8 / PG8 / PH8 / PJ8 |
| 9 | PA9 / PB9 / PC9 / PD9 / PE9 / PF9 / PG9 / PH9 / PJ9 |
| 10 | PA10 / PB10 / PC10 / PD10 / PE10 / PF10 / PG10 / PH10 / PJ10 |
| 11 | PB11 / PC11 / PD11 / PE11 / PF11 / PG11 / PH11 / PJ11 |
| 12 | PB12 / PC12 / PD12 / PE12 / PF12 / PG12 / PH12 |
| 13 | PA13 / PB13 / PC13 / PD13 / PE13 / PF13 / PG13 / PH13 |
| 14 | PA14 / PB14 / PC14 / PD14 / PE14 / PF14 / PG14 / PH14 |
| 15 | PA15 / PB15 / PC15 / PD15 / PE15 / PF15 / PG15 / PH15 |
| 16 | VAVD, LVD and VOVD |
| 17 | RTC alarm |

| EXTI line number | Source |
|---|---|
| 18 | RTC tamper and timestamp event, LXTAL clock stuck |
| 19 | RTC wakeup |
| 20 | CMP0 output |
| 21 | CMP1 output |
| 22 | Ethernet1 wakeup |
| 23 | Ethernet0 wakeup |
| 24 | CAN0 wakeup |
| 25 | CAN1 wakeup |
| 26 | CAN2 wakeup |
| 27 | USART0 wakeup |
| 28 | USART1 wakeup |
| 29 | USART2 wakeup |
| 30 | USART5 wakeup |
| 31 | USBHS0 wakeup |
| 32 | USBHS1 wakeup |
| 33 | I2C0 wakeup |
| 34 | I2C1 wakeup |
| 35 | I2C2 wakeup |
| 36 | I2C3 wakeup |
| 37 | LPDTS wakeup |

**Table 8-5. EXTI source (For GD32H75E)**

| EXTI line number | Source |
|---|---|
| 0 | PA0 / PB0 / PC0 / PD0 / PE0 / PH0 |
| 1 | PA1 / PB1 / PC1 / PD1 / PE1 / PH1 |
| 2 | PA2 / PB2 / PC2 / PD2 / PE2 / PH2 |
| 3 | PA3 / PB3 / PC3 / PD3 / PE3 / PH3 |
| 4 | PA4 / PB4 / PC4 / PD4 / PE4 / PH4 |
| 5 | PA5 / PB5 / PC5 / PD5 / PE5 / PF5 / PH5 |
| 6 | PA6 / PB6 / PC6 / PD6 / PE6 / PF6 / PG6 / PH6 |
| 7 | PA7 / PB7 / PC7 / PD7 / PE7 / PF7 / PG7 / PH7 |
| 8 | PA8 / PB8 / PC8 / PD8 / PE8 / PF8 / PG8 / PH8 |
| 9 | PA9 / PB9 / PC9 / PD9 / PE9 / PF9 / PG9 / PH9 |
| 10 | PA10 / PB10 / PC10 / PD10 / PE10 / PF10 / PH10 |
| 11 | PA11 / PB11 / PC11 / PD11 / PE11 / PF11 / PH11 |
| 12 | PA12 / PB12 / PC12 / PD12 / PE12 / PF12 / PH12 |
| 13 | PA13 / PB13 / PC13 / PD13 / PE13 / PF13 / PG13 / PH13 |
| 14 | PA14 / PB14 / PC14 / PD14 / PE14 / PF14 / PG14 |
| 15 | PA15 / PB15 / PC15 / PD15 / PE15 / PF15 / PG15 |
| 16 | VAVD, LVD and VOVD |

| EXTI line number | Source |
|---|---|
| 17 | RTC alarm |
| 18 | RTC tamper and timestamp event, LXTAL clock stuck |
| 19 | RTC wakeup |
| 20 | CMP0 output |
| 21 | CMP1 output |
| 22 | Reserved |
| 23 | Reserved |
| 24 | CAN0 wakeup |
| 25 | CAN1 wakeup |
| 26 | CAN2 wakeup |
| 27 | USART0 wakeup |
| 28 | USART1 wakeup |
| 29 | USART2 wakeup |
| 30 | USART5 wakeup |
| 31 | USBHS0 wakeup |
| 32 | USBHS1 wakeup |
| 33 | I2C0 wakeup |
| 34 | I2C1 wakeup |
| 35 | I2C2 wakeup |
| 36 | I2C3 wakeup |
| 37 | LPDTS wakeup |

## 8.6. Register definition

EXTI base address: 0x5800 0000

### 8.6.1. Interrupt enable register 0 (EXTI_INTEN0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INTEN31 | INTEN30 | INTEN29 | INTEN28 | INTEN27 | INTEN26 | INTEN25 | INTEN24 | INTEN23 | INTEN22 | INTEN21 | INTEN20 | INTEN19 | INTEN18 | INTEN17 | INTEN16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEN15 | INTEN14 | INTEN13 | INTEN12 | INTEN11 | INTEN10 | INTEN9 | INTEN8 | INTEN7 | INTEN6 | INTEN5 | INTEN4 | INTEN3 | INTEN2 | INTEN1 | INTEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | INTENx | Interrupt enable bit x (x = 0…31)<br>0: Interrupt from linex is disabled.<br>1: Interrupt from linex is enabled. |

### 8.6.2. Event enable register 0 (EXTI_EVEN0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EVEN31 | EVEN30 | EVEN29 | EVEN28 | EVEN27 | EVEN26 | EVEN25 | EVEN24 | EVEN23 | EVEN22 | EVEN21 | EVEN20 | EVEN19 | EVEN18 | EVEN17 | EVEN16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EVEN15 | EVEN14 | EVEN13 | EVEN12 | EVEN11 | EVEN10 | EVEN9 | EVEN8 | EVEN7 | EVEN6 | EVEN5 | EVEN4 | EVEN3 | EVEN2 | EVEN1 | EVEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | EVENx | Event enable bit x (x = 0…31)<br>0: Event from linex is disabled<br>1: Event from linex is enabled |

### 8.6.3. Rising edge trigger enable register 0 (EXTI_RTEN0)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTEN31 | RTEN30 | RTEN29 | RTEN28 | RTEN27 | RTEN26 | RTEN25 | RTEN24 | RTEN23 | RTEN22 | RTEN21 | RTEN20 | RTEN19 | RTEN18 | RTEN17 | RTEN16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RTEN15 | RTEN14 | RTEN13 | RTEN12 | RTEN11 | RTEN10 | RTEN9 | RTEN8 | RTEN7 | RTEN6 | RTEN5 | RTEN4 | RTEN3 | RTEN2 | RTEN1 | RTEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RTENx | Rising edge trigger enable bit x (x = 0…31)<br>0: Rising edge of linex is invalid<br>1: Rising edge of linex is valid as an interrupt / event request |

## 8.6.4. Falling edge trigger enable register 0 (EXTI_FTEN0)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FTEN31 | FTEN30 | FTEN29 | FTEN28 | FTEN27 | FTEN26 | FTEN25 | FTEN24 | FTEN23 | FTEN22 | FTEN21 | FTEN20 | FTEN19 | FTEN18 | FTEN17 | FTEN16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FTEN15 | FTEN14 | FTEN13 | FTEN12 | FTEN11 | FTEN10 | FTEN9 | FTEN8 | FTEN7 | FTEN6 | FTEN5 | FTEN4 | FTEN3 | FTEN2 | FTEN1 | FTEN0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FTENx | Falling edge trigger enable bit x (x = 0…31)<br>0: Falling edge of linex is invalid<br>1: Falling edge of linex is valid as an interrupt / event request |

## 8.6.5. Software interrupt event register 0 (EXTI_SWIEV0)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWIEV31 | SWIEV30 | SWIEV29 | SWIEV28 | SWIEV27 | SWIEV26 | SWIEV25 | SWIEV24 | SWIEV23 | SWIEV22 | SWIEV21 | SWIEV20 | SWIEV19 | SWIEV18 | SWIEV17 | SWIEV16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWIEV15 | SWIEV14 | SWIEV13 | SWIEV12 | SWIEV11 | SWIEV10 | SWIEV9 | SWIEV8 | SWIEV7 | SWIEV6 | SWIEV5 | SWIEV4 | SWIEV3 | SWIEV2 | SWIEV1 | SWIEV0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SWIEVx | Interrupt / event software trigger bit x (x = 0…31)<br>0: Deactivate the EXTIx software interrupt / event request<br>1: Activate the EXTIx software interrupt / event request |

### 8.6.6. Pending register 0 (EXTI_PD0)

Address offset: 0x14

Reset value: 0xXXXX XXXX where X is undefined.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PD31 | PD30 | PD29 | PD28 | PD27 | PD26 | PD25 | PD24 | PD23 | PD22 | PD21 | PD20 | PD19 | PD18 | PD17 | PD16 |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PD15 | PD14 | PD13 | PD12 | PD11 | PD10 | PD9 | PD8 | PD7 | PD6 | PD5 | PD4 | PD3 | PD2 | PD1 | PD0 |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PDx | Interrupt pending status bit x (x = 0…31)<br>0: EXTI linex is not triggered<br>1: EXTI linex is triggered. This bit is cleared to 0 by writing 1 to it. |

### 8.6.7. Interrupt enable register 1 (EXTI_INTEN1)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | INTEN37 | INTEN36 | INTEN35 | INTEN34 | INTEN33 | INTEN32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | INTENx | Interrupt enable bit x (x = 32…37)<br>0: Interrupt from linex is disabled.<br>1: Interrupt from linex is enabled. |

### 8.6.8. Event enable register 1 (EXTI_EVEN1)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | EVEN37 | EVEN36 | EVEN35 | EVEN34 | EVEN33 | EVEN32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | EVENx | Event enable bit x (x = 32…37)<br>0: Event from linex is disabled.<br>1: Event from linex is enabled. |

### 8.6.9. Rising edge trigger enable register 1 (EXTI_RTEN1)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | RTEN37 | RTEN36 | RTEN35 | RTEN34 | RTEN33 | RTEN32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | RTENx | Rising edge trigger enable bit x (x = 32…37)<br>0: Rising edge of linex is invalid<br>1: Rising edge of linex is valid as an interrupt / event request |

### 8.6.10. Falling edge trigger enable register 1 (EXTI_FTEN1)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | FTEN37 | FTEN36 | FTEN35 | FTEN34 | FTEN33 | FTEN32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | FTENx | Falling edge trigger enable bit x (x = 32…37)<br>0: Falling edge of linex is invalid<br>1: Falling edge of linex is valid as an interrupt / event request |

## 8.6.11. Software interrupt event register 1 (EXTI_SWIEV1)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | SWIEV37 | SWIEV36 | SWIEV35 | SWIEV34 | SWIEV33 | SWIEV32 |
| | | | | | | | | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | SWIEVx | Interrupt / event software trigger bit x (x = 32…37)<br>0: Deactivate the EXTIx software interrupt / event request<br>1: Activate the EXTIx software interrupt / event request |

## 8.6.12. Pending register 1 (EXTI_PD1)

Address offset: 0x2C

Reset value: 0x0000 00XX where X is undefined.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | PD37 | PD36 | PD35 | PD34 | PD33 | PD32 |

rc_w1 rc_w1 rc_w1 rc_w1 rc_w1 rc_w1

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | PDx | Interrupt pending status bit x (x = 32…37)<br>0: EXTI linex is not triggered<br>1: EXTI linex is triggered. This bit is cleared to 0 by writing 1 to it. |

# 9. Trigger selection controller (TRIGSEL)

## 9.1. Overview

The trigger selection controller (TRIGSEL) allows software to select the trigger input signal for various peripherals. TRIGSEL provides a flexible mechanism for a peripheral to select different trigger inputs.

With TRIGSEL, there are up to 150 trigger input signals could be selected. Configure the corresponding register to select the different trigger signal for the specified trigger input of each peripheral.

## 9.2. Characteristics

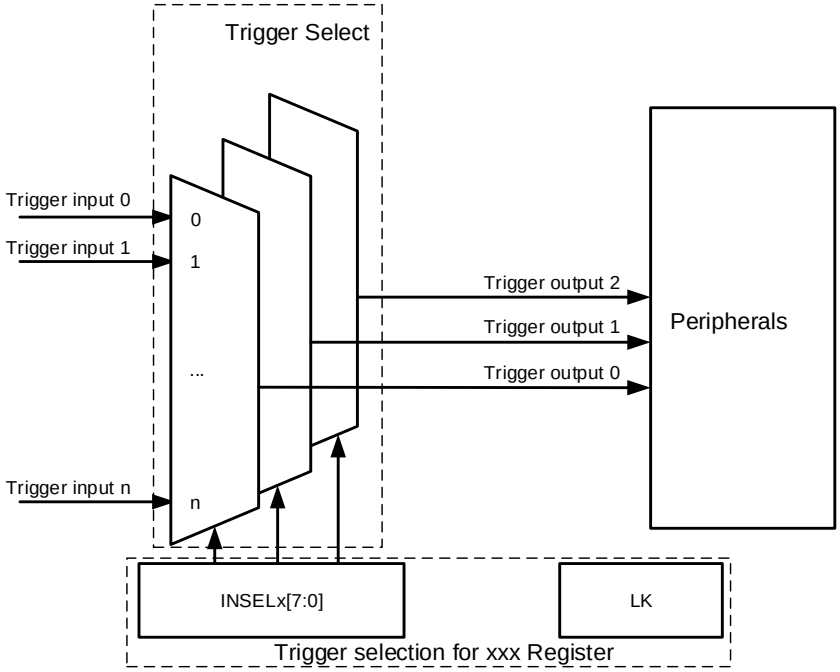
- Support up to 150 trigger input signals.
- Each peripheral has its corresponding register to select trigger input signal.
- Trigger input source could be external input signal or output of peripheral.
- Trigger selection output could be for external output or peripheral input.

## 9.3. Function overview

With TRIGSEL, peripherals that support trigger source selection have dedicated registers to select the trigger input source. Each register can be configured with 3 outputs, which are connected to the trigger input of the peripheral. Each output can select different trigger input sources.

The ***Figure 9-1. TRIGSEL main composition example*** shows the main composition of TRIGSEL.

**Figure 9-1. TRIGSEL main composition example**




## 9.4. Internal connect

The TRIGSEL allows software to select the trigger input for peripherals. The ***Table 9-1. Trigger input bit fields selection*** shows the trigger input register selection.

**Table 9-1. Trigger input bit fields selection**

| fields | bits value | trigger input selection |
|---|---|---|
| INSELx | 0x00 | 0 |
| | 0x01 | 1 |
| | 0x02 | TRIGSEL_IN0 |
| | 0x03 | TRIGSEL_IN1 |
| | 0x04 | TRIGSEL_IN2 |
| | 0x05 | TRIGSEL_IN3 |
| | 0x06 | TRIGSEL_IN4 |
| | 0x07 | TRIGSEL_IN5 |
| | 0x08 | TRIGSEL_IN6 |
| | 0x09 | TRIGSEL_IN7 |
| | 0x0a | TRIGSEL_IN8 |
| | 0x0b | TRIGSEL_IN9 |
| | 0x0c | TRIGSEL_IN10 |
| | 0x0d | TRIGSEL_IN11 |
| | 0x0e | TRIGSEL_IN12 |
| | 0x0f | TRIGSEL_IN13 |
| | 0x10 | LXTAL_TRG |

| fields | bits value | trigger input selection |
|---|---|---|
| | 0x11 | TIMER0_TRGO0 |
| | 0x12 | TIMER0_TRGO1 |
| | 0x13 | TIMER0_CH0 |
| | 0x14 | TIMER0_CH1 |
| | 0x15 | TIMER0_CH2 |
| | 0x16 | TIMER0_CH3 |
| | 0x17 | TIMER0_MCH0 |
| | 0x18 | TIMER0_MCH1 |
| | 0x19 | TIMER0_MCH2 |
| | 0x1a | TIMER0_MCH3 |
| | 0x1b~0x20 | Reserved |
| | 0x21 | TIMER0_BRKIN0 |
| | 0x22 | TIMER0_BRKIN1 |
| | 0x23 | TIMER0_BRKIN2 |
| | 0x24 | TIMER0_ETI |
| | 0x25 | TIMER1_TRGO0 |
| | 0x26 | TIMER1_CH0 |
| | 0x27 | TIMER1_CH1 |
| | 0x28 | TIMER1_CH2 |
| | 0x29 | TIMER1_CH3 |
| | 0x2a | TIMER1_ETI |
| | 0x2b | TIMER2_TRGO0 |
| | 0x2c | TIMER2_CH0 |
| | 0x2d | TIMER2_CH1 |
| | 0x2e | TIMER2_CH2 |
| | 0x2f | TIMER2_CH3 |
| | 0x30 | TIMER2_ETI |
| | 0x31 | TIMER3_TRGO0 |
| | 0x32 | TIMER3_CH0 |
| | 0x33 | TIMER3_CH1 |
| | 0x34 | TIMER3_CH2 |
| | 0x35 | TIMER3_CH3 |
| | 0x36 | TIMER3_ETI |
| | 0x37 | TIMER4_TRGO0 |
| | 0x38 | TIMER4_CH0 |
| | 0x39 | TIMER4_CH1 |
| | 0x3a | TIMER4_CH2 |
| | 0x3b | TIMER4_CH3 |
| | 0x3c | TIMER4_ETI |
| | 0x3d | TIMER5_TRGO0 |
| | 0x3e | TIMER6_TRGO0 |

| fields | bits value | trigger input selection |
|---|---|---|
| | 0x3f | TIMER7_TRGO0 |
| | 0x40 | TIMER7_TRGO1 |
| | 0x41 | TIMER7_CH0 |
| | 0x42 | TIMER7_CH1 |
| | 0x43 | TIMER7_CH2 |
| | 0x44 | TIMER7_CH3 |
| | 0x45 | TIMER7_MCH0 |
| | 0x46 | TIMER7_MCH1 |
| | 0x47 | TIMER7_MCH2 |
| | 0x48 | TIMER7_MCH3 |
| | 0x49~0x4e | Reserved |
| | 0x4f | TIMER7_BRKIN0 |
| | 0x50 | TIMER7_BRKIN1 |
| | 0x51 | TIMER7_BRKIN2 |
| | 0x52 | TIMER7_ETI |
| | 0x53 | TIMER14_TRGO0 |
| | 0x54 | TIMER14_CH0 |
| | 0x55 | TIMER14_CH1 |
| | 0x56 | TIMER14_MCH0 |
| | 0x57~0x58 | Reserved |
| | 0x59 | TIMER14_BRKIN0 |
| | 0x5a | TIMER15_CH0 |
| | 0x5b | TIMER15_MCH0 |
| | 0x5c~0x5d | Reserved |
| | 0x5e | TIMER15_BRKIN0 |
| | 0x5f | TIMER16_CH0 |
| | 0x60 | TIMER16_MCH0 |
| | 0x61~0x62 | Reserved |
| | 0x63 | TIMER16_BRKIN0 |
| | 0x64 | TIMER22_TRGO0 |
| | 0x65 | TIMER22_CH0 |
| | 0x66 | TIMER22_CH1 |
| | 0x67 | TIMER22_CH2 |
| | 0x68 | TIMER22_CH3 |
| | 0x69 | TIMER22_ETI |
| | 0x6a | TIMER23_TRGO0 |
| | 0x6b | TIMER23_CH0 |
| | 0x6c | TIMER23_CH1 |
| | 0x6d | TIMER23_CH2 |
| | 0x6e | TIMER23_CH3 |
| | 0x6f | TIMER23_ETI |

| fields | bits value | trigger input selection |
|---|---|---|
| | 0x70 | TIMER30_TRGO0 |
| | 0x71 | TIMER30_CH0 |
| | 0x72 | TIMER30_CH1 |
| | 0x73 | TIMER30_CH2 |
| | 0x74 | TIMER30_CH3 |
| | 0x75 | TIMER30_ETI |
| | 0x76 | TIMER31_TRGO0 |
| | 0x77 | TIMER31_CH0 |
| | 0x78 | TIMER31_CH1 |
| | 0x79 | TIMER31_CH2 |
| | 0x7a | TIMER31_CH3 |
| | 0x7b | TIMER31_ETI |
| | 0x7c | TIMER40_TRGO0 |
| | 0x7d | TIMER40_CH0 |
| | 0x7e | TIMER40_CH1 |
| | 0x7f | TIMER40_MCH0 |
| | 0x80~0x81 | Reserved |
| | 0x82 | TIMER40_BRKIN0 |
| | 0x83 | TIMER41_TRGO0 |
| | 0x84 | TIMER41_CH0 |
| | 0x85 | TIMER41_CH1 |
| | 0x86 | TIMER41_MCH0 |
| | 0x87~0x88 | Reserved |
| | 0x89 | TIMER41_BRKIN0 |
| | 0x8a | TIMER42_TRGO0 |
| | 0x8b | TIMER42_CH0 |
| | 0x8c | TIMER42_CH1 |
| | 0x8d | TIMER42_MCH0 |
| | 0x8e~0x8f | Reserved |
| | 0x90 | TIMER42_BRKIN0 |
| | 0x91 | TIMER43_TRGO0 |
| | 0x92 | TIMER43_CH0 |
| | 0x93 | TIMER43_CH1 |
| | 0x94 | TIMER43_MCH0 |
| | 0x95~0x96 | Reserved |
| | 0x97 | TIMER43_BRKIN0 |
| | 0x98 | TIMER44_TRGO0 |
| | 0x99 | TIMER44_CH0 |
| | 0x9a | TIMER44_CH1 |
| | 0x9b | TIMER44_MCH0 |
| | 0x9c~0x9d | Reserved |

| fields | bits value | trigger input selection |
|---|---|---|
| | 0x9e | TIMER44_BRKIN0 |
| | 0x9f | TIMER50_TRGO0 |
| | 0xa0 | TIMER51_TRGO0 |
| | 0xa1 | RTC_Alarm |
| | 0xa2 | RTC_TPTS |
| | 0xa3 | ADC0_WD0_OUT |
| | 0xa4 | ADC0_WD1_OUT |
| | 0xa5 | ADC0_WD2_OUT |
| | 0xa6 | ADC1_WD0_OUT |
| | 0xa7 | ADC1_WD1_OUT |
| | 0xa8 | ADC1_WD2_OUT |
| | 0xa9 | ADC2_WD0_OUT |
| | 0xaa | ADC2_WD1_OUT |
| | 0xab | ADC2_WD2_OUT |
| | 0xac | CMP0_OUT |
| | 0xad | CMP1_OUT |
| | 0xae | SAI0_FS0 |
| | 0xaf | SAI0_FS1 |
| | 0xb0 | SAI2_FS0 |
| | 0xb1 | SAI2_FS1 |
| | 0xb2~0xff | Reserved |

*Table 9-2. TRIGSEL input and output mapping* shows the connection relationship between TRIGSEL input and output. Through the INSELx[7:0] bits of TRIGSEL register, an input trigger source can be selected for the output of TRIGSEL. Each TRIGSEL register can configure with up to 3 outputs, which are connected to the corresponding peripherals.

**Table 9-2. TRIGSEL input and output mapping**

| Trigger Source | Trigger select | TRIGSEL Register | TRIGSEL output | Peripherals |
|---|---|---|---|---|
| 0<br>1<br>TRIGSEL_IN0<br>TRIGSEL_IN1<br>TRIGSEL_IN2<br>TRIGSEL_IN3<br>TRIGSEL_IN4<br>TRIGSEL_IN5<br>TRIGSEL_IN6<br>TRIGSEL_IN7<br>TRIGSEL_IN8<br>TRIGSEL_IN9 | INSELx[7:0] | TRIGSEL_EXTOUT_0 | Output0<br>Output1 | TRIGSEL_OUT0<br>TRIGSEL_OUT1 |
| | | TRIGSEL_EXTOUT_1 | Output0<br>Output1 | TRIGSEL_OUT2<br>TRIGSEL_OUT3 |
| | | TRIGSEL_EXTOUT_2 | Output0<br>Output1 | TRIGSEL_OUT4<br>TRIGSEL_OUT5 |

| Trigger Source | Trigger select | TRIGSEL Register | TRIGSEL output | Peripherals |
|---|---|---|---|---|
| TRIGSEL_IN10<br>TRIGSEL_IN11<br>TRIGSEL_IN12<br>TRIGSEL_IN13<br>LXTAL_TRG<br>TIMER0_TRGO0<br>TIMER0_TRGO1<br>TIMER0_CH0<br>TIMER0_CH1<br>TIMER0_CH2<br>TIMER0_CH3<br>TIMER0_MCH0<br>TIMER0_MCH1<br>TIMER0_MCH2<br>TIMER0_MCH3<br>TIMER0_BRKIN0<br>TIMER0_BRKIN1<br>TIMER0_BRKIN2<br>TIMER0_ETI<br>TIMER1_TRGO0<br>TIMER1_CH0<br>TIMER1_CH1<br>TIMER1_CH2<br>TIMER1_CH3<br>TIMER1_ETI<br>TIMER2_TRGO0<br>TIMER2_CH0<br>TIMER2_CH1<br>TIMER2_CH2<br>TIMER2_CH3<br>TIMER2_ETI<br>TIMER3_TRGO0<br>TIMER3_CH0<br>TIMER3_CH1<br>TIMER3_CH2<br>TIMER3_CH3<br>TIMER3_ETI<br>TIMER4_TRGO0<br>TIMER4_CH0<br>TIMER4_CH1 | | TRIGSEL_EXTOUT_3 | Output0<br>Output1 | TRIGSEL_OUT6<br>TRIGSEL_OUT7 |
| | | TRIGSEL_ADC0 | Output0 | ADC0_ROUTRG |
| | | TRIGSEL_ADC1 | Output0 | ADC1_ROUTRG |
| | | TRIGSEL_ADC2 | Output0 | ADC2_ROUTRG |
| | | TRIGSEL_DAC0OUT0 | Output0 | DAC0_OUT0_EXTRG |
| | | TRIGSEL_DAC0OUT1 | Output0 | DAC0_OUT1_EXTRG |
| | | TRIGSEL_TIMER0BRKIN | Output0<br>Output1<br>Output2 | TIMER0_BRKIN0<br>TIMER0_BRKIN1<br>TIMER0_BRKIN2 |
| | | TRIGSEL_TIMER7BRKIN | Output0<br>Output1<br>Output2 | TIMER7_BRKIN0<br>TIMER7_BRKIN1<br>TIMER7_BRKIN2 |
| | | TRIGSEL_TIMER14BRKIN | Output0 | TIMER14_BRKIN0 |
| | | TRIGSEL_TIMER15BRKIN | Output0 | TIMER15_BRKIN0 |

| Trigger Source | Trigger select | TRIGSEL Register | TRIGSEL output | Peripherals |
|---|---|---|---|---|
| TIMER4_CH2<br>TIMER4_CH3<br>TIMER4_ETI<br>TIMER5_TRGO0<br>TIMER6_TRGO0<br>TIMER7_TRGO0<br>TIMER7_TRGO1<br>TIMER7_CH0<br>TIMER7_CH1<br>TIMER7_CH2<br>TIMER7_CH3<br>TIMER7_MCH0<br>TIMER7_MCH1<br>TIMER7_MCH2<br>TIMER7_MCH3<br>TIMER7_BRKIN0<br>TIMER7_BRKIN1<br>TIMER7_BRKIN2<br>TIMER7_ETI<br>TIMER14_TRGO0<br>TIMER14_CH0<br>TIMER14_CH1<br>TIMER14_MCH0<br>TIMER14_BRKIN0<br>TIMER15_CH0<br>TIMER15_MCH0<br>TIMER15_BRKIN0<br>TIMER16_CH0<br>TIMER16_MCH0<br>TIMER16_BRKIN0<br>TIMER22_TRGO0<br>TIMER22_CH0<br>TIMER22_CH1<br>TIMER22_CH2<br>TIMER22_CH3<br>TIMER22_ETI<br>TIMER23_TRGO0<br>TIMER23_CH0<br>TIMER23_CH1<br>TIMER23_CH2 | | TRIGSEL_TIMER16BRKIN | Output0 | TIMER16_BRKIN0 |
| | | TRIGSEL_TIMER40BRKIN | Output0 | TIMER40_BRKIN0 |
| | | TRIGSEL_TIMER41BRKIN | Output0 | TIMER41_BRKIN0 |
| | | TRIGSEL_TIMER42BRKIN | Output0 | TIMER42_BRKIN0 |
| | | TRIGSEL_TIMER43BRKIN | Output0 | TIMER43_BRKIN0 |
| | | TRIGSEL_TIMER44BRKIN | Output0 | TIMER44_BRKIN0 |
| | | TRIGSEL_CAN0 | Output0 | CAN0_EX_TIME_TICK |
| | | TRIGSEL_CAN1 | Output0 | CAN1_EX_TIME_TICK |
| | | TRIGSEL_CAN2 | Output0 | CAN2_EX_TIME_TICK |
| | | TRIGSEL_LPDTS | Output0 | LPDTS_TRG |

| Trigger Source | Trigger select | TRIGSEL Register | TRIGSEL output | Peripherals |
|---|---|---|---|---|
| TIMER23_CH3<br>TIMER23_ETI<br>TIMER30_TRGO0<br>TIMER30_CH0<br>TIMER30_CH1<br>TIMER30_CH2<br>TIMER30_CH3<br>TIMER30_ETI<br>TIMER31_TRGO0<br>TIMER31_CH0<br>TIMER31_CH1<br>TIMER31_CH2<br>TIMER31_CH3<br>TIMER31_ETI<br>TIMER40_TRGO0<br>TIMER40_CH0<br>TIMER40_CH1<br>TIMER40_MCH0<br>TIMER40_BRKIN0<br>TIMER41_TRGO0<br>TIMER41_CH0<br>TIMER41_CH1<br>TIMER41_MCH0<br>TIMER41_BRKIN0<br>TIMER42_TRGO0<br>TIMER42_CH0<br>TIMER42_CH1<br>TIMER42_MCH0<br>TIMER42_BRKIN0<br>TIMER43_TRGO0<br>TIMER43_CH0<br>TIMER43_CH1<br>TIMER43_MCH0<br>TIMER43_BRKIN0<br>TIMER44_TRGO0<br>TIMER44_CH0<br>TIMER44_CH1<br>TIMER44_MCH0<br>TIMER44_BRKIN0<br>TIMER50_TRGO0 | | TRIGSEL_EDOUT | Output0 | EDOUT_TRG |
| | | TRIGSEL_HPDF | Output0 | HPDF_ITRG |
| | | TRIGSEL_TIMER0ETI | Output0 | TIMER0_ETI |
| | | TRIGSEL_TIMER1ETI | Output0 | TIMER1_ETI |
| | | TRIGSEL_TIMER2ETI | Output0 | TIMER2_ETI |
| | | TRIGSEL_TIMER3ETI | Output0 | TIMER3_ETI |
| | | TRIGSEL_TIMER4ETI | Output0 | TIMER4_ETI |
| | | TRIGSEL_TIMER7ETI | Output0 | TIMER7_ETI |
| | | TRIGSEL_TIMER22ETI | Output0 | TIMER22_ETI |
| | | TRIGSEL_TIMER23ETI | Output0 | TIMER23_ETI |

| Trigger Source | Trigger select | TRIGSEL Register | TRIGSEL output | Peripherals |
|---|---|---|---|---|
| TIMER51_TRGO0<br>RTC_Alarm<br>RTC_TPTS<br>ADC0_WD0_OUT<br>ADC0_WD1_OUT<br>ADC0_WD2_OUT<br>ADC1_WD0_OUT<br>ADC1_WD1_OUT<br>ADC1_WD2_OUT<br>ADC2_WD0_OUT<br>ADC2_WD1_OUT<br>ADC2_WD2_OUT<br>CMP0_OUT<br>CMP1_OUT<br>SAI0_FS0<br>SAI0_FS1<br>SAI2_FS0<br>SAI2_FS1 | | TRIGSEL_TIMER30ETI | Output0 | TIMER30_ETI |
| | | TRIGSEL_TIMER31ETI | Output0 | TIMER31_ETI |
| | | TRIGSEL_TIMER0ITI14 | Output0 | TIMER0_ITI14 |
| | | TRIGSEL_TIMER1ITI14 | Output0 | TIMER1_ITI14 |
| | | TRIGSEL_TIMER2ITI14 | Output0 | TIMER2_ITI14 |
| | | TRIGSEL_TIMER3ITI14 | Output0 | TIMER3_ITI14 |
| | | TRIGSEL_TIMER4ITI14 | Output0 | TIMER4_ITI14 |
| | | TRIGSEL_TIMER7ITI14 | Output0 | TIMER7_ITI14 |
| | | TRIGSEL_TIMER14ITI14 | Output0 | TIMER14_ITI14 |
| | | TRIGSEL_TIMER22ITI14 | Output0 | TIMER22_ITI14 |

| Trigger Source | Trigger select | TRIGSEL Register | TRIGSEL output | Peripherals |
|---|---|---|---|---|
| | | TRIGSEL_TIMER23ITI14 | Output0 | TIMER23_ITI14 |
| | | TRIGSEL_TIMER30ITI14 | Output0 | TIMER30_ITI14 |
| | | TRIGSEL_TIMER31ITI14 | Output0 | TIMER31_ITI14 |
| | | TRIGSEL_TIMER40ITI14 | Output0 | TIMER40_ITI14 |
| | | TRIGSEL_TIMER41ITI14 | Output0 | TIMER41_ITI14 |
| | | TRIGSEL_TIMER42ITI14 | Output0 | TIMER42_ITI14 |
| | | TRIGSEL_TIMER43ITI14 | Output0 | TIMER43_ITI14 |
| | | TRIGSEL_TIMER44ITI14 | Output0 | TIMER44_ITI14 |

**Note:** TIMERx_BRKINx can only select TIMERx_BRKINx as trigger. TIMERx_ITI14 cannot select CMPx_OUT ,LXTAL_TRG, TIMERx_BRKINx and their own signals as trigger input. Other output can select all input as trigger source except TIMERx_BRKINx. When illegal data is selected for these outputs, the output will be selected as 0.

When trigger input selection INSELx[7:0] bits is configured to be 0x00, TRIGSEL trigger input is selected as low level; when configured to be 0x01, TRIGSEL trigger input is selected as high level.

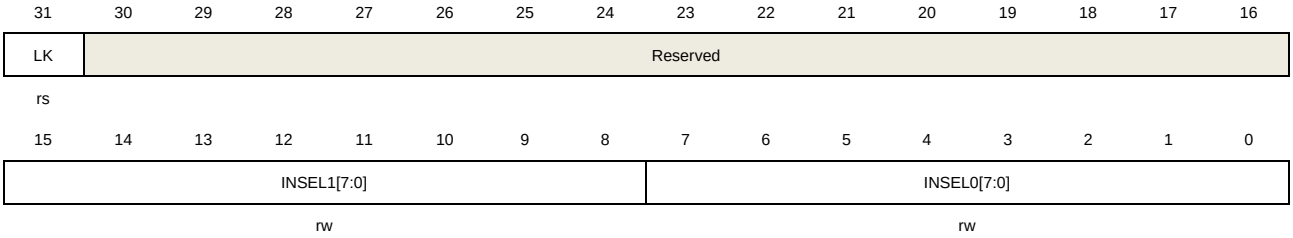
## 9.5. Register definition

TRIGSEL base address: 0x4001 8400

### 9.5.1. Trigger selection for EXTOUT register 0 (TRIGSEL_EXTOUT_0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

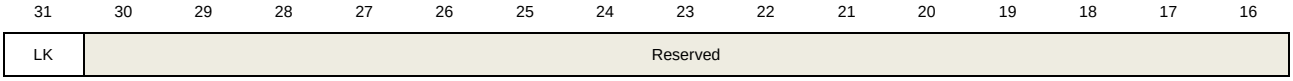



| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_EXTOUT_0 register.<br>0: TRIGSEL_EXTOUT_0 register write is enabled.<br>1: TRIGSEL_EXTOUT_0 register write is disabled. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | INSEL1[7:0] | Trigger input source selection for output1<br>These bits are used to select trigger input signal connected to output1. The output can be as the source of TRIGSEL_OUT1 (external output1 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TRIGSEL_OUT0 (external output0 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.2. Trigger selection for EXTOUT register 1 (TRIGSEL_EXTOUT_1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSEL1[7:0] | | | | | | | | INSEL0[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_EXTOUT_1 register.<br>0: TRIGSEL_EXTOUT_1 register write is enabled.<br>1: TRIGSEL_EXTOUT_1 register write is disabled. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | INSEL1[7:0] | Trigger input source selection for output1<br>These bits are used to select trigger input signal connected to output1. The output can be as the source of TRIGSEL_OUT3 (external output3 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TRIGSEL_OUT2 (external output2 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.3. Trigger selection for EXTOUT register 2 (TRIGSEL_EXTOUT_2)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSEL1[7:0] | | | | | | | | INSEL0[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_EXTOUT_2 register.<br>0: TRIGSEL_EXTOUT_2 register write is enabled.<br>1: TRIGSEL_EXTOUT_2 register write is disabled. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | INSEL1[7:0] | Trigger input source selection for output1 |

| | | |
|---|---|---|
| | | These bits are used to select trigger input signal connected to output1. The output can be as the source of TRIGSEL_OUT5 (external output5 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TRIGSEL_OUT4 (external output4 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.4. Trigger selection for EXTOUT register 3 (TRIGSEL_EXTOUT_3)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSEL1[7:0] | | | | | | | | INSEL0[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

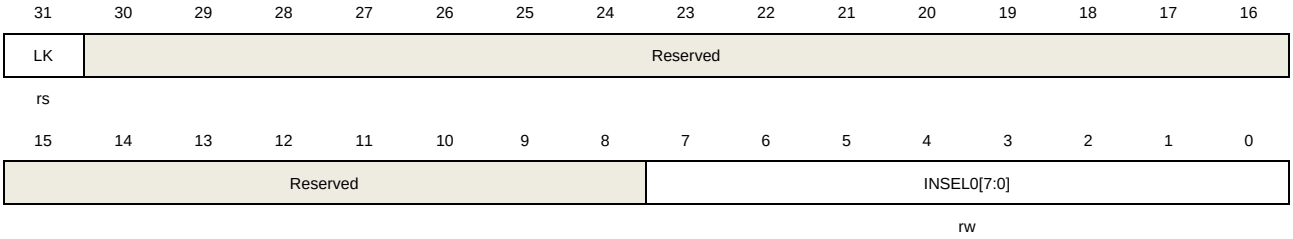
| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_EXTOUT_3 register.<br>0: TRIGSEL_EXTOUT_3 register write is enabled.<br>1: TRIGSEL_EXTOUT_3 register write is disabled. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | INSEL1[7:0] | Trigger input source selection for output1<br>These bits are used to select trigger input signal connected to output1. The output can be as the source of TRIGSEL_OUT7 (external output7 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TRIGSEL_OUT6 (external output6 signal). For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.5. Trigger selection for ADC0 register (TRIGSEL_ADC0)

Address offset: 0x10

Reset value: 0x0000 1113

This register has to be accessed by word (32-bit).



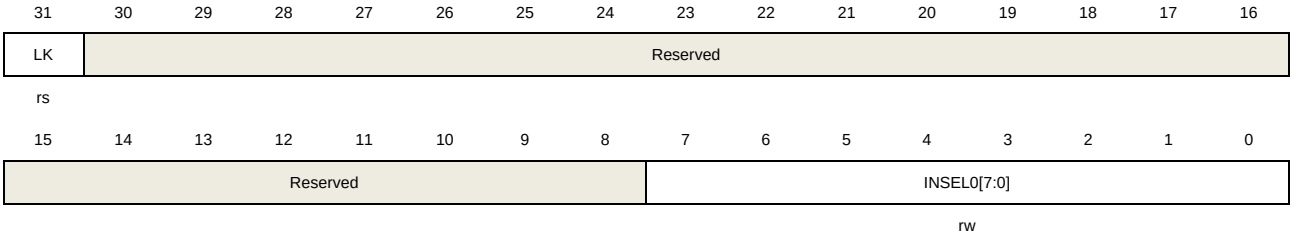

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_ADC0 register.<br>0: TRIGSEL_ADC0 register write is enabled.<br>1: TRIGSEL_ADC0 register write is disabled. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of ADC0_ROUTRG(ADC0 routine sequence) trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.6. Trigger selection for ADC1 register (TRIGSEL_ADC1)

Address offset: 0x14

Reset value: 0x0000 1113

This register has to be accessed by word (32-bit).



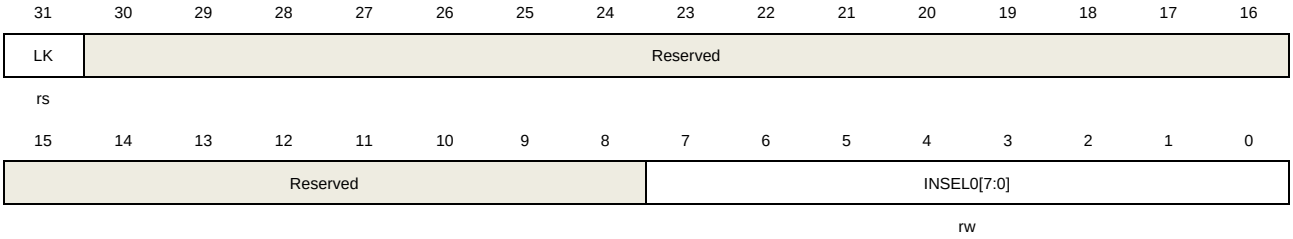

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_ADC1 register.<br>0: TRIGSEL_ADC1 register write is enabled.<br>1: TRIGSEL_ADC1 register write is disabled. |

| | | |
|---|---|---|
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of ADC1_ROUTRG(ADC1 routine sequence) trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.7. Trigger selection for ADC2 register (TRIGSEL_ADC2)

Address offset: 0x18
Reset value: 0x0000 1113

This register has to be accessed by word (32-bit).

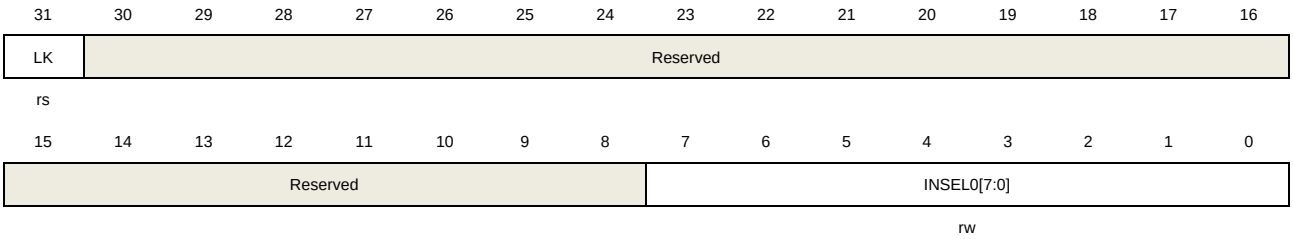



| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_ADC2 register.<br>0: TRIGSEL_ADC2 register write is enabled.<br>1: TRIGSEL_ADC2 register write is disabled. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of ADC2_ROUTRG(ADC2 routine sequence) trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.8. Trigger selection for DAC0_OUT0 register (TRIGSEL_DAC0OUT0)

Address offset: 0x1C
Reset value: 0x0000 0025

This register has to be accessed by word (32-bit).



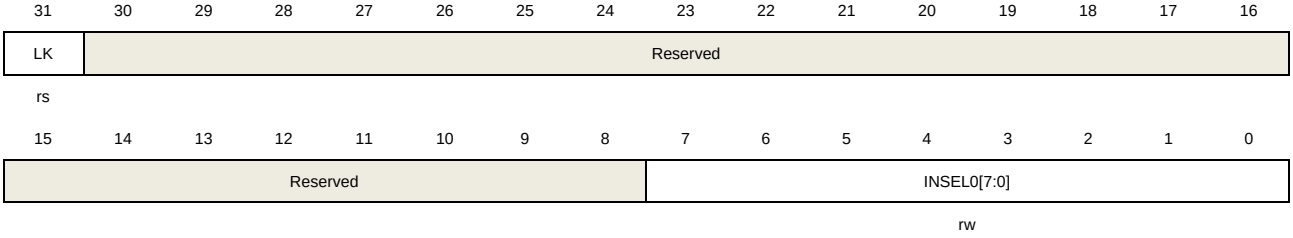

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_DAC0OUT0 register.<br>0: TRIGSEL_DAC0OUT0 register write is enabled.<br>1: TRIGSEL_DAC0OUT0 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of DAC0_OUT0_EXTRIG (DAC0_OUT0 external trigger) input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.9. Trigger selection for DAC0_OUT1 register (TRIGSEL_DAC0OUT1)

Address offset: 0x20

Reset value: 0x0000 0025

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_DAC0OUT1 register.<br>0: TRIGSEL_DAC0OUT1 register write is enabled.<br>1: TRIGSEL_DAC0OUT1 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output |

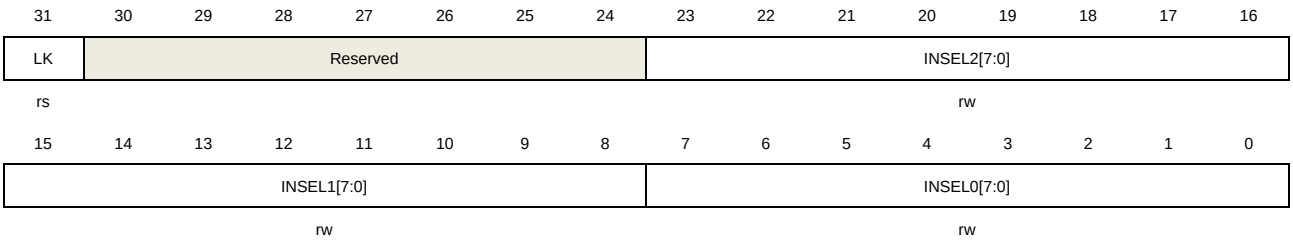
is used as the source of DAC0_OUT1_EXTRIG (DAC0_OUT1 external trigger) input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***.

## 9.5.10. Trigger selection for TIMER0_BRKIN register (TRIGSEL_TIMER0BRKIN)

Address offset: 0x24

Reset value: 0x0023 2221

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER0BRKIN register.<br>0: TRIGSEL_TIMER0BRKIN register write is enabled.<br>1: TRIGSEL_TIMER0BRKIN register write is disabled. |
| 30:24 | Reserved | Must be kept at reset value. |
| 23:16 | INSEL2[7:0] | Trigger input source selection for output2<br>These bits are used to select trigger input signal connected to output2. The output is used as the source of TIMER0_BRKIN2 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 15:8 | INSEL1[7:0] | Trigger input source selection for output1<br>These bits are used to select trigger input signal connected to output1. The output is used as the source of TIMER0_BRKIN1 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER0_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.11. Trigger selection for TIMER7_BRKIN register (TRIGSEL_TIMER7BRKIN)

Address offset: 0x28

Reset value: 0x0051 504F

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | INSEL2[7:0] | | | | | | | |
| rs | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSEL1[7:0] | | | | | | | | INSEL0[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER7BRKIN register.<br>0: TRIGSEL_TIMER7BRKIN register write is enabled.<br>1: TRIGSEL_TIMER7BRKIN register write is disabled. |
| 30:24 | Reserved | Must be kept at reset value. |
| 23:16 | INSEL2[7:0] | Trigger input source selection for output2<br>These bits are used to select trigger input signal connected to output2. The output is used as the source of TIMER7_BRKIN2 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 15:8 | INSEL1[7:0] | Trigger input source selection for output1<br>These bits are used to select trigger input signal connected to output1. The output is used as the source of TIMER7_BRKIN1 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER7_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.12. Trigger selection for TIMER14_BRKIN register (TRIGSEL_TIMER14BRKIN)

Address offset: 0x2C
Reset value: 0x0000 0059

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

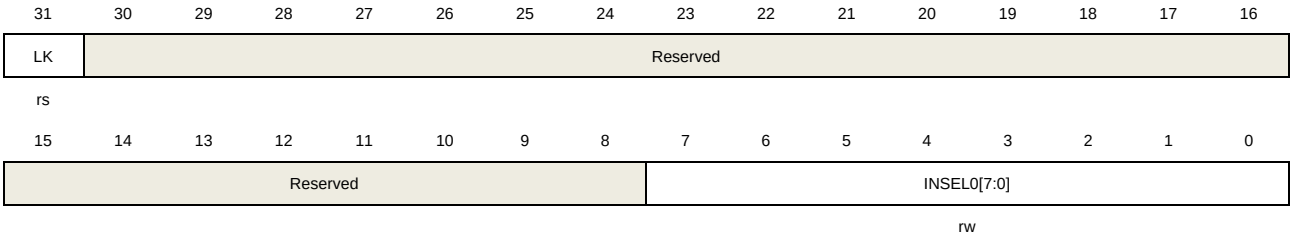
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER14BRKIN register.<br>0: TRIGSEL_TIMER14BRKIN register write is enabled.<br>1: TRIGSEL_TIMER14BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER14_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.13. Trigger selection for TIMER15_BRKIN register (TRIGSEL_TIMER15BRKIN)

Address offset: 0x30
Reset value: 0x0000 005E

This register has to be accessed by word (32-bit).



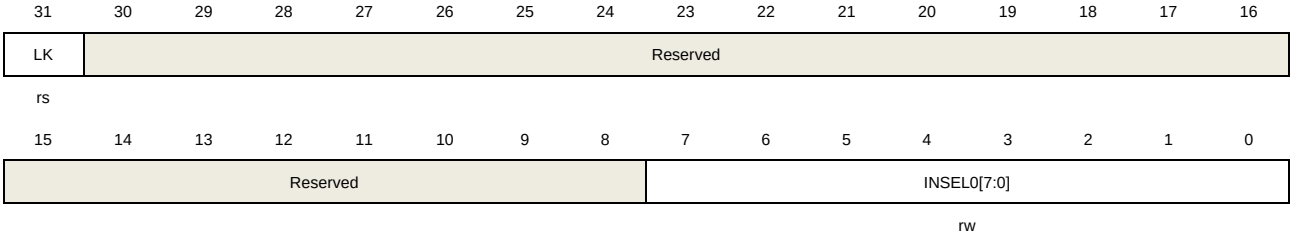

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER15BRKIN register.<br>0: TRIGSEL_TIMER15BRKIN register write is enabled.<br>1: TRIGSEL_TIMER15BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER15_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.14. Trigger selection for TIMER16_BRKIN register (TRIGSEL_TIMER16BRKIN)

Address offset: 0x34

Reset value: 0x0000 0063

This register has to be accessed by word (32-bit).



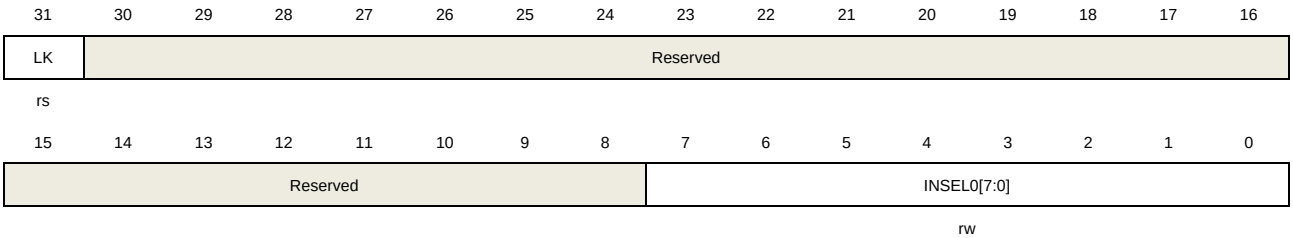

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER16BRKIN register.<br>0: TRIGSEL_TIMER16BRKIN register write is enabled.<br>1: TRIGSEL_TIMER16BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER16_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.15. Trigger selection for TIMER40_BRKIN register (TRIGSEL_TIMER40BRKIN)

Address offset: 0x38

Reset value: 0x0000 0082

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock. |

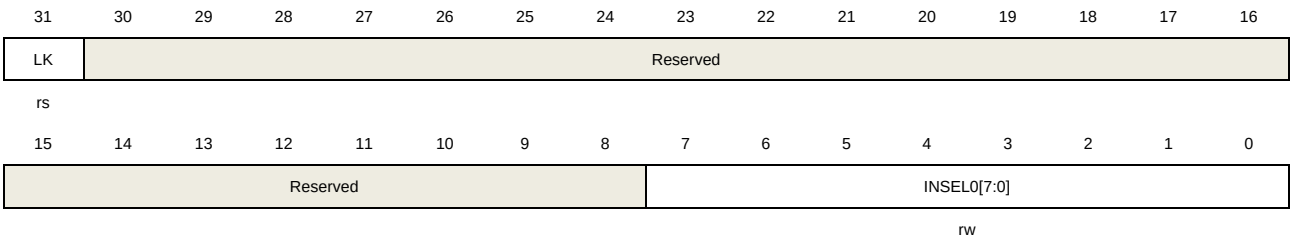
| | | |
|---|---|---|
| | | This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER40BRKIN register.<br>0: TRIGSEL_TIMER40BRKIN register write is enabled.<br>1: TRIGSEL_TIMER40BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER40_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.16. Trigger selection for TIMER41_BRKIN register (TRIGSEL_TIMER41BRKIN)

Address offset: 0x3C

Reset value: 0x0000 0089

This register has to be accessed by word (32-bit).



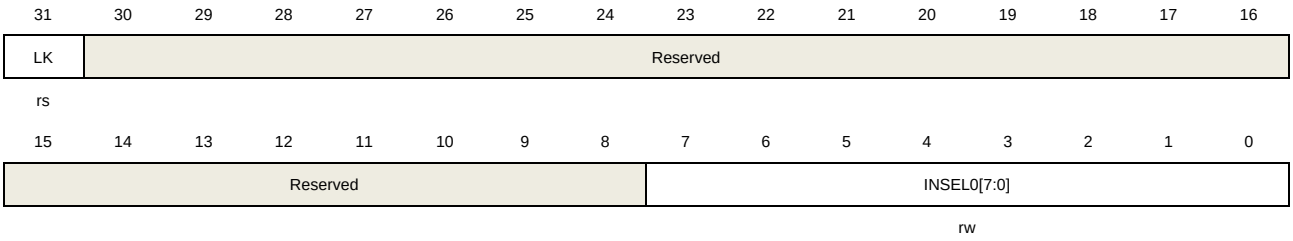

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER41BRKIN register.<br>0: TRIGSEL_TIMER41BRKIN register write is enabled.<br>1: TRIGSEL_TIMER41BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER41_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.17. Trigger selection for TIMER42_BRKIN register (TRIGSEL_TIMER42BRKIN)

Address offset: 0x40

Reset value: 0x0000 0090

This register has to be accessed by word (32-bit).



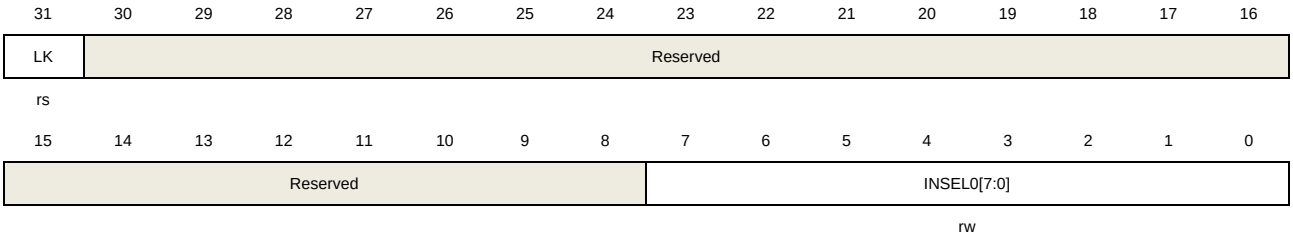

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER42BRKIN register.<br>0: TRIGSEL_TIMER42BRKIN register write is enabled.<br>1: TRIGSEL_TIMER42BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER42_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.18. Trigger selection for TIMER43_BRKIN register (TRIGSEL_TIMER43BRKIN)

Address offset: 0x44

Reset value: 0x0000 0097

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER43BRKIN register.<br>0: TRIGSEL_TIMER43BRKIN register write is enabled.<br>1: TRIGSEL_TIMER43BRKIN register write is disabled. |

| | | |
|---|---|---|
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER43_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.19. Trigger selection for TIMER44_BRKIN register (TRIGSEL_TIMER44BRKIN)

Address offset: 0x48
Reset value: 0x0000 009e

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER44BRKIN register.<br>0: TRIGSEL_TIMER44BRKIN register write is enabled.<br>1: TRIGSEL_TIMER44BRKIN register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER44_BRKIN0 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.20. Trigger selection for CAN0 register (TRIGSEL_CAN0)

Address offset: 0x4C
Reset value: 0x0000 003d

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_CAN0 register.<br>0: TRIGSEL_CAN0 register write is enabled.<br>1: TRIGSEL_CAN0 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of CAN0_EX_TIME_TICK trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.21. Trigger selection for CAN1 register (TRIGSEL_CAN1)

Address offset: 0x50
Reset value: 0x0000 003d

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_CAN1 register.<br>0: TRIGSEL_CAN1 register write is enabled.<br>1: TRIGSEL_CAN1 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of CAN1_EX_TIME_TICK trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.22. Trigger selection for CAN2 register (TRIGSEL_CAN2)

Address offset: 0x54

Reset value: 0x0000 003d

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_CAN2 register.<br>0: TRIGSEL_CAN2 register write is enabled.<br>1: TRIGSEL_CAN2 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of CAN2_EX_TIME_TICK trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.23. Trigger selection for LPDTS register (TRIGSEL_LPDTS)

Address offset: 0x58

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_LPDTS register.<br>0: TRIGSEL_LPDTS register write is enabled. |

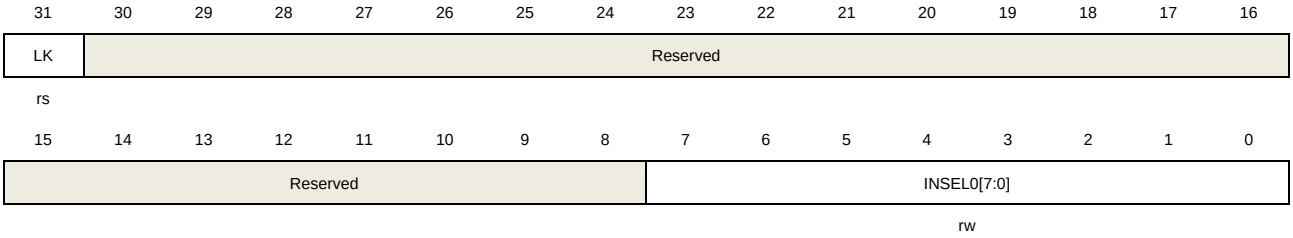
| | | |
|---|---|---|
| | | 1: TRIGSEL_LPDTS register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of LPDTS_TRG trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.24. Trigger selection for TIMER0_ETI register (TRIGSEL_TIMER0ETI)

Address offset: 0x5C

Reset value: 0x0000 0024

This register has to be accessed by word (32-bit).



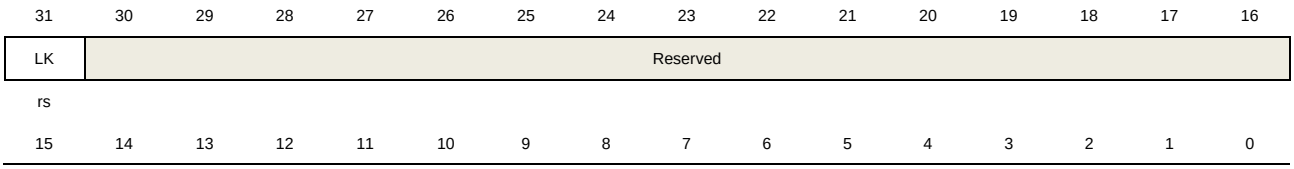

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER0ETI register.<br>0: TRIGSEL_TIMER0ETI register write is enabled.<br>1: TRIGSEL_TIMER0ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER0_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.25. Trigger selection for TIMER1_ETI register (TRIGSEL_TIMER1ETI)

Address offset: 0x60

Reset value: 0x0000 002a

This register has to be accessed by word (32-bit).



| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER1ETI register.<br>0: TRIGSEL_TIMER1ETI register write is enabled.<br>1: TRIGSEL_TIMER1ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER1_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.26. Trigger selection for TIMER2_ETI register (TRIGSEL_TIMER2ETI)

Address offset: 0x64
Reset value: 0x0000 0030

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER2ETI register.<br>0: TRIGSEL_TIMER2ETI register write is enabled.<br>1: TRIGSEL_TIMER2ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER2_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.27. Trigger selection for TIMER3_ETI register (TRIGSEL_TIMER3ETI)

Address offset: 0x68

Reset value: 0x0000 0036

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER3ETI register.<br>0: TRIGSEL_TIMER3ETI register write is enabled.<br>1: TRIGSEL_TIMER3ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER3_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.28. Trigger selection for TIMER4_ETI register (TRIGSEL_TIMER4ETI)

Address offset: 0x6C

Reset value: 0x0000 003c

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER4ETI register.<br>0: TRIGSEL_TIMER4ETI register write is enabled. |

| | | |
|---|---|---|
| | | 1: TRIGSEL_TIMER4ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER4_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.29. Trigger selection for TIMER7_ETI register (TRIGSEL_TIMER7ETI)

Address offset: 0x70

Reset value: 0x0000 0052

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER7ETI register.<br>0: TRIGSEL_TIMER7ETI register write is enabled.<br>1: TRIGSEL_TIMER7ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER7_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.30. Trigger selection for TIMER22_ETI register (TRIGSEL_TIMER22ETI)

Address offset: 0x74

Reset value: 0x0000 0069

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER22ETI register.<br>0: TRIGSEL_TIMER22ETI register write is enabled.<br>1: TRIGSEL_TIMER22ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER22_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.31. Trigger selection for TIMER23_ETI register (TRIGSEL_TIMER23ETI)

Address offset: 0x78
Reset value: 0x0000 006f

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER23ETI register.<br>0: TRIGSEL_TIMER23ETI register write is enabled.<br>1: TRIGSEL_TIMER23ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER23_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.32. Trigger selection for TIMER30_ETI register (TRIGSEL_TIMER30ETI)

Address offset: 0x7C

Reset value: 0x0000 0075

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER30ETI register.<br>0: TRIGSEL_TIMER30ETI register write is enabled.<br>1: TRIGSEL_TIMER30ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER30_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.33. Trigger selection for TIMER31_ETI register (TRIGSEL_TIMER31ETI)

Address offset: 0x80

Reset value: 0x0000 007b

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER31ETI register.<br>0: TRIGSEL_TIMER31ETI register write is enabled. |

| | | |
|---|---|---|
| | | 1: TRIGSEL_TIMER31ETI register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER31_ETI trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.34. Trigger selection for EDOUT register (TRIGSEL_EDOUT)

Address offset: 0x84

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_EDOUT register.<br>0: TRIGSEL_EDOUT register write is enabled.<br>1: TRIGSEL_EDOUT register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the souce of EDOUT_TRG trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection*** |

## 9.5.35. Trigger selection for HPDF register (TRIGSEL_HPDF)

Address offset: 0x88

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_HPDF register.<br>0: TRIGSEL_HPDF register write is enabled.<br>1: TRIGSEL_HPDF register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of HPDF_ITRG trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.36. Trigger selection for TIMER0_ITI14 register (TRIGSEL_TIMER0ITI14)

Address offset: 0x8C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

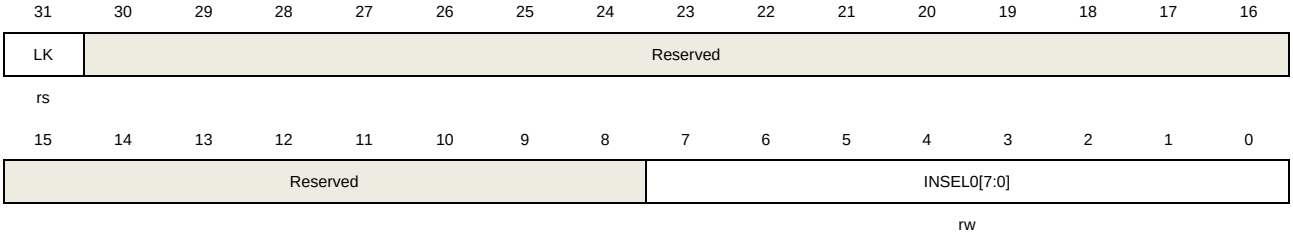
| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER0ITI14 register.<br>0: TRIGSEL_TIMER0ITI14 register write is enabled.<br>1: TRIGSEL_TIMER0ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER0_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.37. Trigger selection for TIMER1_ITI14 register (TRIGSEL_TIMER1ITI14)

Address offset: 0x90

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



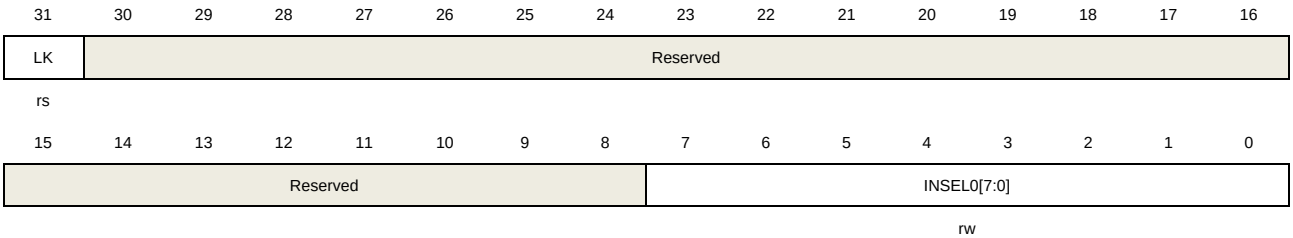

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER1ITI14 register.<br>0: TRIGSEL_TIMER1ITI14 register write is enabled.<br>1: TRIGSEL_TIMER1ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER1_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.38. Trigger selection for TIMER2_ITI14 register (TRIGSEL_TIMER2ITI14)

Address offset: 0x94

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER2ITI14 register.<br>0: TRIGSEL_TIMER2ITI14 register write is enabled. |

| | | |
|---|---|---|
| | | 1: TRIGSEL_TIMER2ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER2_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.39. Trigger selection for TIMER3_ITI14 register (TRIGSEL_TIMER3ITI14)

Address offset: 0x98

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER3ITI14 register.<br>0: TRIGSEL_TIMER3ITI14 register write is enabled.<br>1: TRIGSEL_TIMER3ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER3_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.40. Trigger selection for TIMER4_ITI14 register (TRIGSEL_TIMER4ITI14)

Address offset: 0x9C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER4ITI14 register.<br>0: TRIGSEL_TIMER4ITI14 register write is enabled.<br>1: TRIGSEL_TIMER4ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER4_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.41. Trigger selection for TIMER7_ITI14 register (TRIGSEL_TIMER7ITI14)

Address offset: 0xA0
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER7ITI14 register.<br>0: TRIGSEL_TIMER7ITI14 register write is enabled.<br>1: TRIGSEL_TIMER7ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER7_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.42. Trigger selection for TIMER14_ITI14 register (TRIGSEL_TIMER14ITI14)

Address offset: 0xA4

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER14ITI14 register.<br>0: TRIGSEL_TIMER14ITI14 register write is enabled.<br>1: TRIGSEL_TIMER14ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER14_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

### 9.5.43. Trigger selection for TIMER22_ITI14 register (TRIGSEL_TIMER22ITI14)

Address offset: 0xA8

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER22ITI14 register.<br>0: TRIGSEL_TIMER22ITI14 register write is enabled. |

| | | |
|---|---|---|
| | | 1: TRIGSEL_TIMER22ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER22_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.44. Trigger selection for TIMER23_ITI14 register (TRIGSEL_TIMER23ITI14)

Address offset: 0xAC

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER23ITI14 register.<br>0: TRIGSEL_TIMER23ITI14 register write is enabled.<br>1: TRIGSEL_TIMER23ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER23_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.45. Trigger selection for TIMER30_ITI14 register (TRIGSEL_TIMER30ITI14)

Address offset: 0xB0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER30ITI14 register.<br>0: TRIGSEL_TIMER30ITI14 register write is enabled.<br>1: TRIGSEL_TIMER30ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER30_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.46. Trigger selection for TIMER31_ITI14 register (TRIGSEL_TIMER31ITI14)

Address offset: 0xB4
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER31ITI14 register.<br>0: TRIGSEL_TIMER31ITI14 register write is enabled.<br>1: TRIGSEL_TIMER31ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER31_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.47. Trigger selection for TIMER40_ITI14 register (TRIGSEL_TIMER40ITI14)

Address offset: 0xB8

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER40ITI14 register.<br>0: TRIGSEL_TIMER40ITI14 register write is enabled.<br>1: TRIGSEL_TIMER40ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER40_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.48. Trigger selection for TIMER41_ITI14 register (TRIGSEL_TIMER41ITI14)

Address offset: 0xBC

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER41ITI14 register.<br>0: TRIGSEL_TIMER41ITI14 register write is enabled. |

| | | |
|---|---|---|
| | | 1: TRIGSEL_TIMER41ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER41_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.49. Trigger selection for TIMER42_ITI14 register (TRIGSEL_TIMER42ITI14)

Address offset: 0xC0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER42ITI14 register.<br>0: TRIGSEL_TIMER42ITI14 register write is enabled.<br>1: TRIGSEL_TIMER42ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER42_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.50. Trigger selection for TIMER43_ITI14 register (TRIGSEL_TIMER43ITI14)

Address offset: 0xC4

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | INSEL0[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER43ITI14 register.<br>0: TRIGSEL_TIMER43ITI14 register write is enabled.<br>1: TRIGSEL_TIMER43ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER43_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

## 9.5.51. Trigger selection for TIMER44_ITI14 register (TRIGSEL_TIMER44ITI14)

Address offset: 0xC8
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LK | Reserved | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | INSEL0[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | TRIGSEL register lock.<br>This bit is set by software and cleared only by a system reset. When it is set, it disables write access to TRIGSEL_TIMER44ITI14 register.<br>0: TRIGSEL_TIMER44ITI14 register write is enabled.<br>1: TRIGSEL_TIMER44ITI14 register write is disabled. |
| 30:8 | Reserved | Must be kept at reset value. |
| 7:0 | INSEL0[7:0] | Trigger input source selection for output0<br>These bits are used to select trigger input signal connected to output0. The output is used as the source of TIMER44_ITI14 trigger input. For the detailed configuration, please refer to ***Table 9-1. Trigger input bit fields selection***. |

# 10. General-purpose and alternate-function I/Os (GPIO and AFIO)

## 10.1. Overview

For GD32H73x_75x series there are up to 131 general purpose I/O pins (GPIO), named PA0~PA10, PA13~PA15, PB0~PB13, PC0~PC15, PD0~PD15, PE0~PE15, PF0~PF15, PG0~PG15, PH0~PH15, PJ8~PJ11, PK0~PK2 for the device to implement logic input/output functions. Each GPIO port has related control and configuration registers to satisfy the requirements of specific applications. The external interrupts on the GPIO pins of the device have related control and configuration registers in the Interrupt/Event Controller Unit (EXTI).

The GPIO ports are pin-shared with other alternative functions (AFs) to obtain maximum flexibility on the package pins. The GPIO pins can be used as alternative functional pins by configuring the corresponding registers regardless of the AF input or output pins.

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), input, peripheral alternate function or analog mode. Each GPIO pin can be configured as pull-up, pull-down or no pull-up/pull-down. All GPIOs are high-current capable except for analog mode.

## 10.2. Characteristics

- Input/output direction control.
- Schmitt trigger input function enable control.
- Each pin weak pull-up/pull-down function.
- Output push-pull/open drain enable control.
- Output set/reset control.
- External interrupt with programmable trigger edge – using EXTI configuration registers.
- Analog input/output configuration.
- Alternate function input/output configuration.
- Port configuration lock.
- Single cycle toggle output capability.

## 10.3. Function overview

Each of the general-purpose I/O ports can be configured as GPIO inputs, GPIO outputs, AF function or analog mode by GPIO 32-bit configuration registers (GPIOx_CTL). When select AF function, the pad input or output is decided by selected AF function output enable. When the port is output (GPIO output or AFIO output), it can be configured as push-pull or open drain mode by GPIO output mode registers (GPIOx_OMODE). And the port max speed can be configured by GPIO output speed registers (GPIOx_OSPD). Each port can be configured

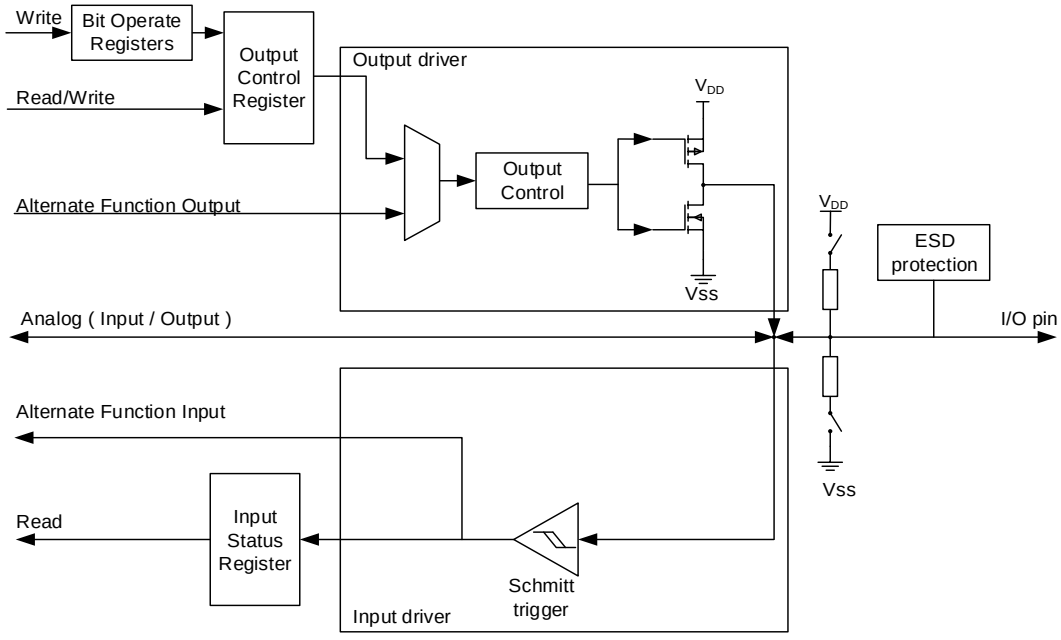
as floating (no pull-up and pull-down), pull-up or pull-down function by GPIO pull-up/pull-down registers (GPIOx_PUD).

**Table 10-1. GPIO configuration table**

| PAD TYPE | | | CTLy | OMy | PUDy |
|---|---|---|---|---|---|
| GPIO INPUT | X | Floating | 00 | X | 00 |
| | | pull-up | | | 01 |
| | | pull-down | | | 10 |
| GPIO OUTPUT | push-pull | Floating | 01 | 0 | 00 |
| | | pull-up | | | 01 |
| | | pull-down | | | 10 |
| | open-drain | Floating | | 1 | 00 |
| | | pull-up | | | 01 |
| | | pull-down | | | 10 |
| AFIO INPUT | X | Floating | 10 | X | 00 |
| | | pull-up | | | 01 |
| | | pull-down | | | 10 |
| AFIO OUTPUT | push-pull | Floating | 10 | 0 | 00 |
| | | pull-up | | | 01 |
| | | pull-down | | | 10 |
| | open-drain | Floating | | 1 | 00 |
| | | pull-up | | | 01 |
| | | pull-down | | | 10 |
| ANALOG | X | X | 11 | X | XX |

*Figure 10-1. Basic structure of a standard I/O port bit* shows the basic structure of an I/O port bit.

**Figure 10-1. Basic structure of a standard I/O port bit**



### 10.3.1. GPIO pin configuration

During or just after the reset period, the alternative functions are all inactive and the GPIO ports (except ports A/B) are configured into the analog mode that input disabled without Pull-Up(PU)/Pull-Down(PD) resistors. But the JTAG/Serial-Wired Debug pins are in input PU/PD mode after reset:

PA15: JTDI in PU mode
PA14: JTCK / SWCLK in PD mode
PA13: JTMS / SWDIO in PU mode
PB4: NJTRST in PU mode
PB3: JTDO in output mode

The GPIO pins can be configured as inputs or outputs. When the GPIO pins are configured as input pins, all GPIO pins have an internal weak pull-up and weak pull-down which can be chosen. And the data on the external pins can be captured at every AHB clock cycle to the port input status register (GPIOx_ISTAT).

When the GPIO pins are configured as output pins, user can configure the speed of the ports. And chooses the output driver mode: Push-Pull or Open-Drain mode. The value of the port output control register (GPIOx_OCTL) is output on the I/O pin.

There is no need to read-then-write when programming the GPIOx_OCTL at bit level, the user can modify only one or several bits in a single atomic AHB write access by programming ‘1’ to the bit operate register (GPIOx_BOP, or for clearing only GPIOx_BC, or for toggle only GPIOx_TG). The other bits will not be affected.

### 10.3.2. External interrupt/event lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured as input mode.

### 10.3.3. Alternate functions (AF)

When the port is configured as AFIO (set CTLy bits to “0b10”, which is in GPIOx_CTL registers), the port is used as peripheral alternate functions. Each port has sixteen alternate functions can be configured by GPIO alternate functions selected registers (GPIOx_AFSELy (y = 0,1)). The detail alternate function assignments for each port are in the device datasheet.

### 10.3.4. Additional functions

Some pins have additional functions, which have priority over the configuration in the standard GPIO registers. When for ADC, DAC, CMP or additional functions, the port must be configured as analog mode. When for RTC, WKUPx and oscillators additional functions, the port type is set automatically by related RTC, PMU and RCU registers. These ports can be used as normal GPIO when the additional functions disabled.

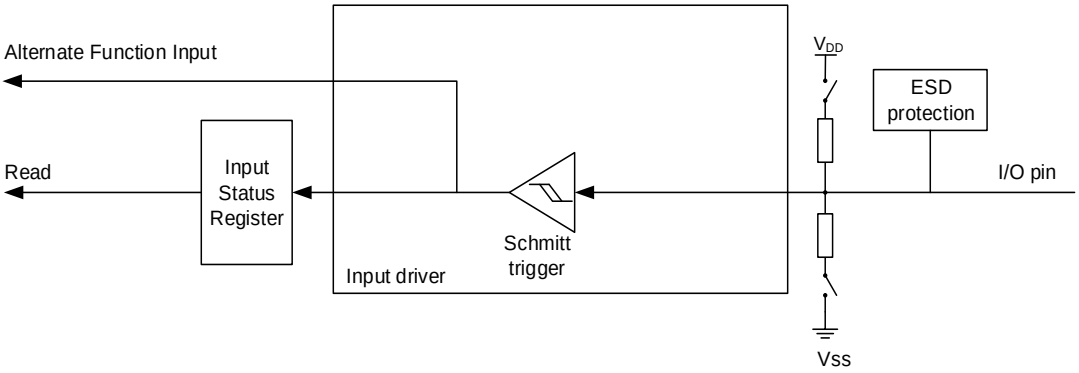
### 10.3.5. Input configuration

When GPIO pin is configured as input:

- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors could be chosen.
- Every AHB clock cycle the data present on the I/O pin is got to the port input status register.
- The output buffer is disabled.

*Figure 10-2. Input configuration* shows the input configuration.

**Figure 10-2. Input configuration**



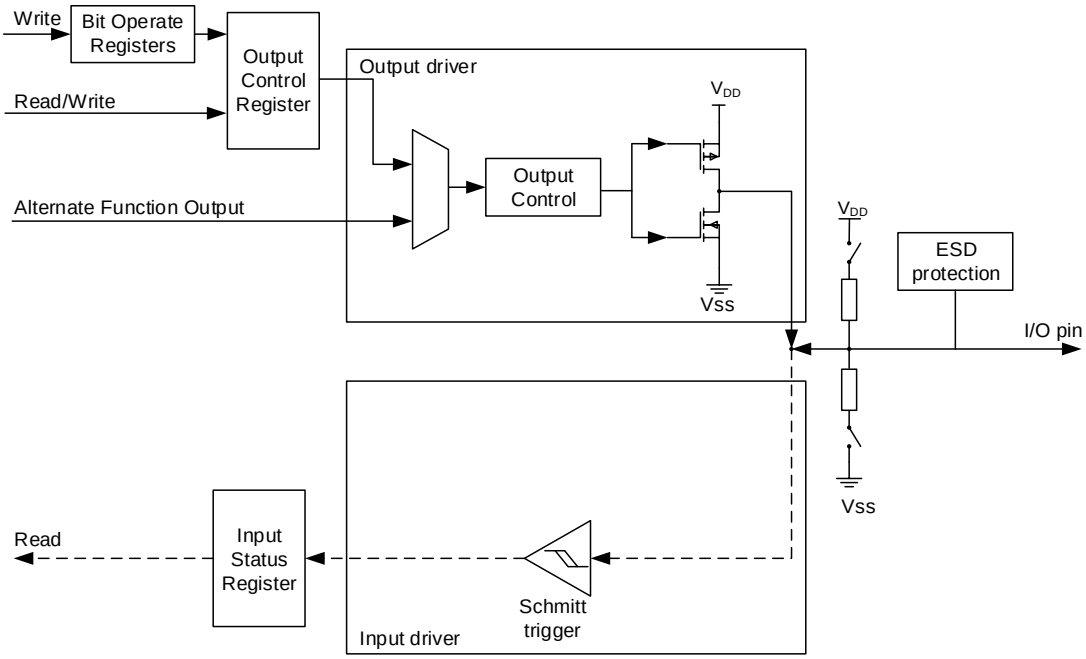

### 10.3.6. Output configuration

When GPIO pin is configured as output:

- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors could be chosen.
- The output buffer is enabled.
- Open Drain Mode: The pad output low level when a “0” in the output control register; while the pad leaves Hi-Z when a “1” in the output control register.
- Push-Pull Mode: The pad output low level when a “0” in the output control register; while the pad output high level when a “1” in the output control register.
- A read access to the port output control register gets the last written value.
- A read access to the port input status register gets the I/O state.

*Figure 10-3. Output configuration* shows the output configuration.

**Figure 10-3. Output configuration**




## 10.3.7. Analog configuration

When GPIO pin is used as analog configuration:

- The weak pull-up and pull-down resistors are disabled.
- The output buffer is disabled.
- The schmitt trigger input is disabled.
- The port input status register of this I/O port bit is “0”.

*Figure 10-4. Analog configuration* shows the analog configuration.

**Figure 10-4. Analog configuration**




## 10.3.8. Alternate function (AF) configuration

To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

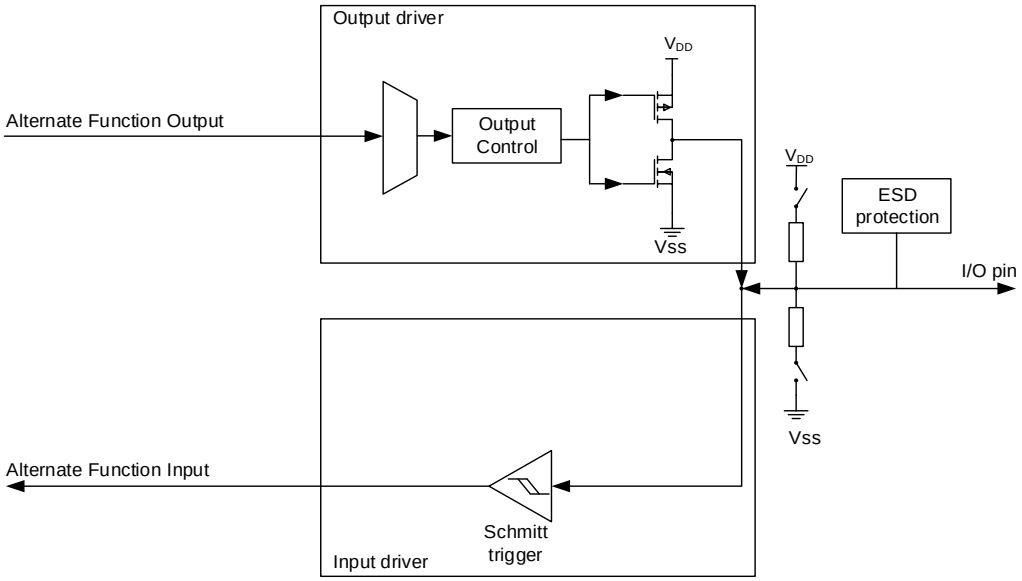
When be configured as alternate function:

- The output buffer is enabled in open-drain or push-pull configuration.
- The output buffer is driven by the peripheral.
- The schmitt trigger input is enabled.

- The weak pull-up and pull-down resistors could be chosen.
- The I/O pin data is stored into the port input status register every AHB clock.
- A read access to the port input status register gets the I/O state.
- A read access to the port output control register gets the last written value.

*Figure 10-5. Alternate function configuration* shows the alternate function configuration.

**Figure 10-5. Alternate function configuration**




## 10.3.9. GPIO locking function

The locking mechanism allows the IO configuration to be protected.

The protected registers are GPIOx_CTL, GPIOx_OMODE, GPIOx_OSPD, GPIOx_PUD and GPIOx_AFSELy (y=0, 1). It allows the I/O configuration to be frozen by the 32-bit locking register (GPIOx_LOCK). When the special LOCK sequence has occurred on LKK bit in GPIOx_LOCK register and the LKy bit is set in GPIOx_LOCK register, the corresponding port is locked and the corresponding port configuration cannot be modified until the next reset. It recommended to be used in the configuration of driving a power module.

## 10.3.10. GPIO single cycle toggle function

GPIO could toggle the I/O output level in single AHB cycle by writing 1 to the corresponding bit of GPIOx_TG register. The output signal frequency could up to the half of the AHB clock.

## 10.3.11. I/O compensation unit

The compensation unit is used to control the commutation slew rate (tfall/trise) to reduce I/O noise on the power supply.

This unit provides the best compensation code under conditions such as current temperature and environment. When the CPS_RDY bit of SYSCFG_CPSCTL is set, the compensation code stored in this area can be read. The user can also configure the compensation code by

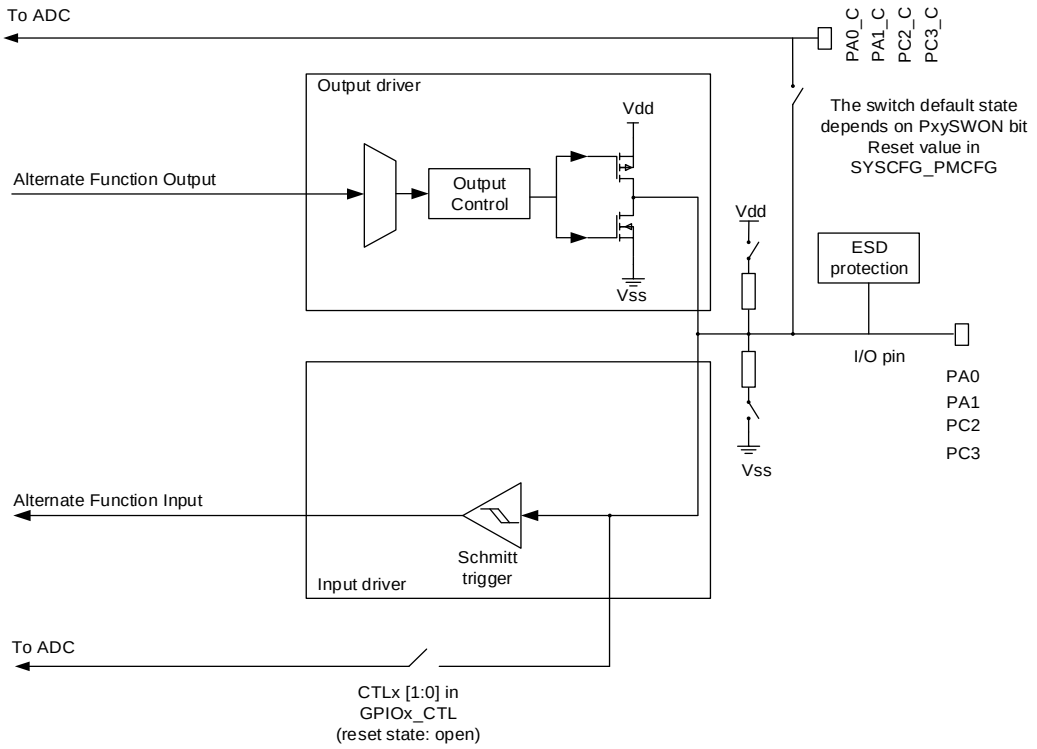
programming the SYSCFG_CPSCCCFG register.

The I/O compensation unit has 2 voltage ranges: 1.62 to 2.0V and 2.7 to 3.6V. Both voltage ranges here refer to the VDD voltage range of the chip.When VDD is 1.62 to 2.0V, the I/O speed is low because the VDD voltage is low. Therefore, set IOSPDOP to 1 can improve the I/O speed. When VDD is 2.7V to 3.6V, the VDD voltage is high enough and the I/O speed is fast. Therefore, IOSPDOP is not required to set.

### 10.3.12. Analog configuration for ADC

For GD32H73x_75x series, PA0_C、PA1_C、PC2_C and PC3_C are connected directly to ADC analog input as show ***Figure 10-6. Analog configuration for ADC for GD32H73x***: Pxy_C and Pxy pins are directly connected through analog switches (refer system register).

**Figure 10-6. Analog configuration for ADC for GD32H73x_75x**




### 10.3.13. Input filtering

The type of input filtering for each GPIO pin can be select by configuring GPIOx_IFTP register. In the case of GPIO, filtering can be specified to synchronize only to CK_AHB or through the sampling window. For pins configured as peripheral input, in addition to synchronization to CK_AHB or through the sampling window, the input can also be asynchronous.

**Note:** The status of the GPIOx_ISTAT register is the signal before GPIO filtering, and the signal output to EXTI is the signal after GPIO filtering.

### Asynchronous input

This mode is used for peripherals that do not need to input synchronization or perform synchronization by the peripheral itself. If the pin is used as GPIO, the asynchronous option is invalid, and the input filter is synchronized to CK_AHB by default.

**Note:** When the peripheral performs synchronization by itself, using input synchronization may result in unexpected results. In this case, the user should ensure that GPIO is configured asynchronously.

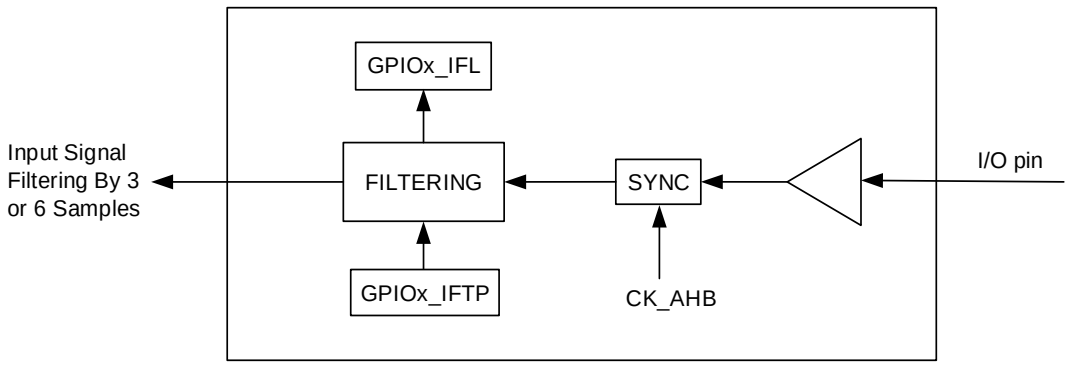
### Synchronization to CK_AHB only

In this mode, the input signal is only associated with CK_AHB synchronization. Since the input signal is asynchronous, it may need a delay period of CK_AHB to change the input of MCU. The signal will not be further filtered after that.

### Filtering using the sampling window

In this mode, the signal first communicates with the system clock (CK_AHB), and then the filtering process through the specified number of cycles will be carry out before allowing the input to be changed. Users need to specify two parameters for this type of filtering: sampling period and sampling times.

**Figure 10-7. Filtering using the sampling window**




#### Samping period

To filtering the signal, the input signal is sampled in a fixed period. The sampling period is specified by the user and determines the duration between samples, or the samping frequency relative to CK_AHB.

The sampling period is determined by FLPRDx in register GPIOx_IFL. The sampling period can be configured as 8 input signal groups. For example, GPIO0 to GPIO7 use FLPRD0, GPIO8 to GPIO15 use FLPRD1.

If FLPRD0 in register GPIOx_IFL is 0, the sampling frequency is $f_{CK_AHB}$. For example, if $f_{CK_AHB}$=100MHz, the signal will be sampled at 100 MHz or every 10ns.

If FLPRD0 in register GPIOx_IFL is 0xFF(255), the sampling frequency is

$f_{CK_AHB}$×1÷(2×FLPRDx). For example, if $f_{CK_AHB}$=100MHz, the signal will be sampled at 100MHz×1÷(2×255) or every 5.1us.

**Sampling times**

The sampling times of signal are 3 samples or 6 samples, detailed description in input filter type register (GPIOx_IFTP). When three or six consecutive cycles are the same, the change of input will be transmitted to MCU.

**Total sampling window width**

The sampling window is the time consumed to sample the input signal, as shown in ***Figure 10-8. Input filtering clock cycle***. The total width of the window can be determined by calculating the sampling period and sampling times

In order for the input filter to detect the change of the input, the signal level must be stable in the width of the sampling window or a longer time.

The number of sampling windows is always one less than the number of samples. For three sampling windows, the width of sampling window is two sampling periods. Similarly, for six sampling windows, the width of sampling window is five sampling periods.

**Note:** External signal variation and sampling period and CK_AHB is asynchronous. Due to the asynchronism of the external signal, the input should be stable for a time greater than the width of the sampling window to ensure that the change of the signal can be detected logically. The extra time required can reach the extra sampling period plus $T_{CK_AHB}$。

**Example of sampling window**

As shown in ***Figure 10-8. Input filtering clock cycle***, the input filtering configuration is as follows:

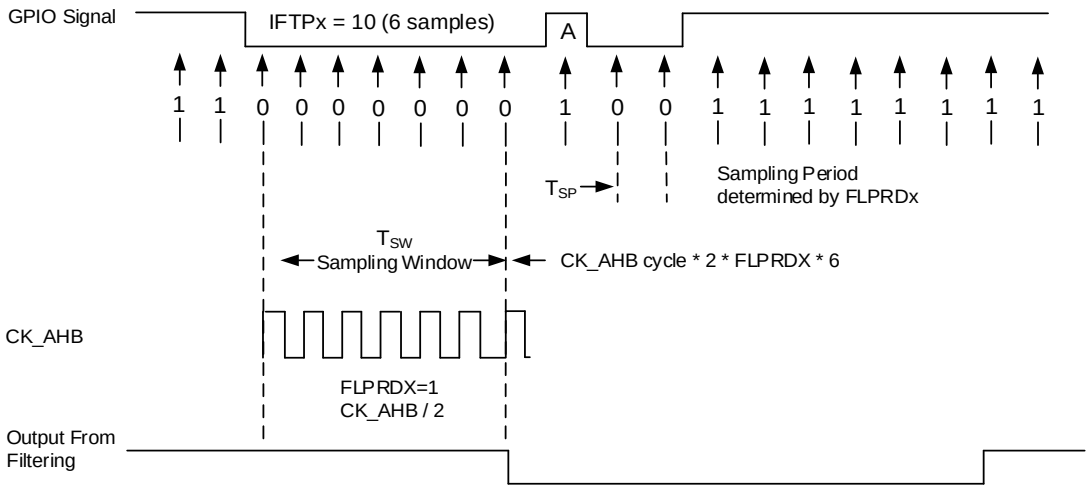
- IFTP0[1:0]=10 in GPIOx_IFTP register, which means there are 6 sampling points;
- FLPRD0=1 in GPIOx_IFL register, sampling period

  $T_{SP}$=2×FLPRD0×$T_{CK_AHB}$=2×$T_{CK_AHB}$;

The configuration results are as follows:

- The sampling width is: $T_{SW}$=6×$T_{SP}$=6×2×FLPRD0×$T_{CK_AHB}$=6×2×$T_{CK_AHB}$;
- If $T_{CK_AHB}$=10ns, the duration of the sampling window is:
  $T_{SP}$=2×$T_{CK_AHB}$=2×10ns=20ns
  $T_{SW}$=6×2×$T_{CK_AHB}$=6×20ns=120ns
- To illustrate the asynchronous nature of the input relative to the sampling period and the system clock, an additional sampling period and CK_AHB cycle may be required to detect the change of input signal.
  $T_{SW}$+$T_{CK_AHB}$=120+10=130ns
- In ***Figure 10-8. Input filtering clock cycle***, the interference (A) is shorter than the total sampling window, so it will be filtered.

**Figure 10-8. Input filtering clock cycle**


GPIO Signal
IFTPx = 10 (6 samples)
A
1 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1
TSP
Sampling Period
determined by FLPRDx
TSW
Sampling Window
CK_AHB cycle * 2 * FLPRDX * 6
CK_AHB
FLPRDX=1
CK_AHB / 2
Output From
Filtering

## 10.4. Register definition

GPIOA base address: 0x5802 0000

GPIOB base address: 0x5802 0400

GPIOC base address: 0x5802 0800

GPIOD base address: 0x5802 0C00

GPIOE base address: 0x5802 1000

GPIOF base address: 0x5802 1400

GPIOG base address: 0x5802 1800

GPIOH base address: 0x5802 1C00

GPIOJ base address: 0x5802 2400

GPIOK base address: 0x5802 2800

### 10.4.1. Port control register (GPIOx_CTL, x=A...H, J, K)

Address offset: 0x00

Reset value: 0xABFF FFFF for port A; 0xFFFF FE8F for port B; 0xFFFF FFFF for others.

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 30 | 29 28 | 27 26 | 25 24 | 23 22 | 21 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|
| CTL15[1:0] | CTL14[1:0] | CTL13[1:0] | CTL12[1:0] | CTL11[1:0] | CTL10[1:0] | CTL9[1:0] | CTL8[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|
| CTL7[1:0] | CTL6[1:0] | CTL5[1:0] | CTL4[1:0] | CTL3[1:0] | CTL2[1:0] | CTL1[1:0] | CTL0[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | CTL15[1:0] | Pin 15 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 29:28 | CTL14[1:0] | Pin 14 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 27:26 | CTL13[1:0] | Pin 13 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 25:24 | CTL12[1:0] | Pin 12 configuration bits |

| | | |
|---|---|---|
| | | These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 23:22 | CTL11[1:0] | Pin 11 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 21:20 | CTL10[1:0] | Pin 10 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 19:18 | CTL9[1:0] | Pin 9 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 17:16 | CTL8[1:0] | Pin 8 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 15:14 | CTL7[1:0] | Pin 7 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 13:12 | CTL6[1:0] | Pin 6 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 11:10 | CTL5[1:0] | Pin 5 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 9:8 | CTL4[1:0] | Pin 4 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 7:6 | CTL3[1:0] | Pin 3 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 5:4 | CTL2[1:0] | Pin 2 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 3:2 | CTL1[1:0] | Pin 1 configuration bits<br>These bits are set and cleared by software.<br>Refer to CTL0[1:0] description |
| 1:0 | CTL0[1:0] | Pin 0 configuration bits<br>These bits are set and cleared by software.<br>00: Input mode |

01: GPIO output mode

10: Alternate function mode

11: Analog mode(reset value)

### 10.4.2. Port output mode register (GPIOx_OMODE, x=A…H, J, K)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OM15 | OM14 | OM13 | OM12 | OM11 | OM10 | OM9 | OM8 | OM7 | OM6 | OM5 | OM4 | OM3 | OM2 | OM1 | OM0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15 | OM15 | Pin 15 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 14 | OM14 | Pin 14 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 13 | OM13 | Pin 13 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 12 | OM12 | Pin 12 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 11 | OM11 | Pin 11 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 10 | OM10 | Pin 10 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 9 | OM9 | Pin 9 output mode bit<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to OM0 description |
| 8 | OM8 | Pin 8 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 7 | OM7 | Pin 7 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 6 | OM6 | Pin 6 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 5 | OM5 | Pin 5 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 4 | OM4 | Pin 4 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 3 | OM3 | Pin 3 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 2 | OM2 | Pin 2 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 1 | OM1 | Pin 1 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM0 description |
| 0 | OM0 | Pin 0 output mode bit<br>These bits are set and cleared by software.<br>0: Output push-pull mode (reset value)<br>1: Output open-drain mode |

### 10.4.3. Port output speed register (GPIOx_OSPD, x=A...H, J, K)

Address offset: 0x08

Reset value: 0x0C00 0000 for port A; 0x0000 00C0 for port B; 0x0000 0000 for others.

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OSPD15[1:0] | | OSPD14[1:0] | | OSPD13[1:0] | | OSPD12[1:0] | | OSPD11[1:0] | | OSPD10[1:0] | | OSPD9[1:0] | | OSPD8[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OSPD7[1:0] | | OSPD6[1:0] | | OSPD5[1:0] | | OSPD4[1:0] | | OSPD3[1:0] | | OSPD2[1:0] | | OSPD1[1:0] | | OSPD0[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | OSPD15[1:0] | Pin 15 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 29:28 | OSPD14[1:0] | Pin 14 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 27:26 | OSPD13[1:0] | Pin 13 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 25:24 | OSPD12[1:0] | Pin 12 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 23:22 | OSPD11[1:0] | Pin 11 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 21:20 | OSPD10[1:0] | Pin 10 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 19:18 | OSPD9[1:0] | Pin 9 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 17:16 | OSPD8[1:0] | Pin 8 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 15:14 | OSPD7[1:0] | Pin 7 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 13:12 | OSPD6[1:0] | Pin 6 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 11:10 | OSPD5[1:0] | Pin 5 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 9:8 | OSPD4[1:0] | Pin 4 output max speed bits |

| | | |
|---|---|---|
| | | These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 7:6 | OSPD3[1:0] | Pin 3 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 5:4 | OSPD2[1:0] | Pin 2 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 3:2 | OSPD1[1:0] | Pin 1 output max speed bits<br>These bits are set and cleared by software.<br>Refer to OSPD0[1:0] description |
| 1:0 | OSPD0[1:0] | Pin 0 output max speed bits<br>These bits are set and cleared by software.<br>00: Output max speed 12M (reset value)<br>01: Output max speed 60M<br>10: Output max speed 85M<br>11: Output max speed 100/220M |

### 10.4.4. Port pull-up/down register (GPIOx_PUD, x=A…H, J, K)

Address offset: 0x0C

Reset value: 0x6400 0000 for port A; 0x0000 0100 for port B; 0x0000 0000 for others.

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 30 | 29 28 | 27 26 | 25 24 | 23 22 | 21 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|
| PUD15[1:0] | PUD14[1:0] | PUD13[1:0] | PUD12[1:0] | PUD11[1:0] | PUD10[1:0] | PUD9[1:0] | PUD8[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |
| **15 14** | **13 12** | **11 10** | **9 8** | **7 6** | **5 4** | **3 2** | **1 0** |
| PUD7[1:0] | PUD6[1:0] | PUD5[1:0] | PUD4[1:0] | PUD3[1:0] | PUD2[1:0] | PUD1[1:0] | PUD0[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | PUD15[1:0] | Pin 15 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 29:28 | PUD14[1:0] | Pin 14 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 27:26 | PUD13[1:0] | Pin 13 pull-up or pull-down bits<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to PUD0[1:0] description |
| 25:24 | PUD12[1:0] | Pin 12 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 23:22 | PUD11[1:0] | Pin 11 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 21:20 | PUD10[1:0] | Pin 10 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 19:18 | PUD9[1:0] | Pin 9 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 17:16 | PUD8[1:0] | Pin 8 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 15:14 | PUD7[1:0] | Pin 7 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 13:12 | PUD6[1:0] | Pin 6 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 11:10 | PUD5[1:0] | Pin 5 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 9:8 | PUD4[1:0] | Pin 4 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 7:6 | PUD3[1:0] | Pin 3 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 5:4 | PUD2[1:0] | Pin 2 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 3:2 | PUD1[1:0] | Pin 1 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD0[1:0] description |
| 1:0 | PUD0[1:0] | Pin 0 pull-up or pull-down bits |

These bits are set and cleared by software.

00: Floating mode, no pull-up and pull-down (reset value)

01: With pull-up mode

10: With pull-down mode

11: Reserved

### 10.4.5. Port input status register (GPIOx_ISTAT, x=A…H, J, K)

Address offset: 0x10

Reset value: 0x0000 XXXX

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ISTAT15 | ISTAT14 | ISTAT13 | ISTAT12 | ISTAT11 | ISTAT10 | ISTAT9 | ISTAT8 | ISTAT7 | ISTAT6 | ISTAT5 | ISTAT4 | ISTAT3 | ISTAT2 | ISTAT1 | ISTAT0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | ISTATy | Port input status (y=0..15)<br>These bits are set and cleared by hardware.<br>0: Input signal low<br>1: Input signal high |

### 10.4.6. Port output control register (GPIOx_OCTL, x=A…H, J, K)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OCTL15 | OCTL14 | OCTL13 | OCTL12 | OCTL11 | OCTL10 | OCTL9 | OCTL8 | OCTL7 | OCTL6 | OCTL5 | OCTL4 | OCTL3 | OCTL2 | OCTL1 | OCTL0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | OCTLy | Port output control (y=0..15) |

These bits are set and cleared by software.

0: Pin output low

1: Pin output high

## 10.4.7. Port bit operate register (GPIOx_BOP, x=A...H, J, K)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BOP15 | BOP14 | BOP13 | BOP12 | BOP11 | BOP10 | BOP9 | BOP8 | BOP7 | BOP6 | BOP5 | BOP4 | BOP3 | BOP2 | BOP1 | BOP0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Cry | Port clear bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Clear the corresponding OCTLy bit |
| 15:0 | BOPy | Port set bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Set the corresponding OCTLy bit |

## 10.4.8. Port configuration lock register (GPIOx_LOCK, x=A...H, J, K)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | LKK |
| | | | | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LK15 | LK14 | LK13 | LK12 | LK11 | LK10 | LK9 | LK8 | LK7 | LK6 | LK5 | LK4 | LK3 | LK2 | LK1 | LK0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 16 | LKK | Lock key<br>It can only be set by using the lock key writing sequence. And is always readable.<br>0: GPIOx_LOCK register and the port configuration are not locked<br>1: GPIOx_LOCK register is locked until an MCU reset<br>LOCK key writing sequence:<br>Write 1→Write 0→Write 1→ Read 0→ Read 1<br>**Note:** The value of LKy(y=0..15) must be held during the LOCK Key writing sequence. |
| 15:0 | LKy | Port lock bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: Port configuration not locked<br>1: Port configuration locked |

### 10.4.9. Alternate function selected register 0 (GPIOx_AFSEL0, x=A...H, J, K)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEL7[3:0] | | | | SEL6[3:0] | | | | SEL5[3:0] | | | | SEL4[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEL3[3:0] | | | | SEL2[3:0] | | | | SEL1[3:0] | | | | SEL0[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | SEL7[3:0] | Pin 7 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 27:24 | SEL6[3:0] | Pin 6 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 23:20 | SEL5[3:0] | Pin 5 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 19:16 | SEL4[3:0] | Pin 4 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 15:12 | SEL3[3:0] | Pin 3 alternate function selected<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to SEL0[3:0] description |
| 11:8 | SEL2[3:0] | Pin 2 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 7:4 | SEL1[3:0] | Pin 1 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL0[3:0] description |
| 3:0 | SEL0[3:0] | Pin 0 alternate function selected<br>These bits are set and cleared by software.<br>0000: AF0 selected (reset value)<br>0001: AF1 selected<br>0010: AF2 selected<br>0011: AF3 selected<br>0100: AF4 selected<br>0101: AF5 selected<br>0110: AF6 selected<br>0111: AF7 selected<br>1000: AF8 selected<br>1001: AF9 selected<br>1010: AF10 selected<br>1011: AF11 selected<br>1100: AF12 selected<br>1101: AF13 selected<br>1110: AF14 selected<br>1111: AF15 selected |

### 10.4.10. Alternate function selected register 1 (GPIOx_AFSEL1, x=A…H, J, K)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEL15[3:0] | | | | SEL14[3:0] | | | | SEL13[3:0] | | | | SEL12[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEL11[3:0] | | | | SEL10[3:0] | | | | SEL9[3:0] | | | | SEL8[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | SEL15[3:0] | Pin 15 alternate function selected<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to SEL8[3:0] description |
| 27:24 | SEL14[3:0] | Pin 14 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 23:20 | SEL13[3:0] | Pin 13 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 19:16 | SEL12[3:0] | Pin 12 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 15:12 | SEL11[3:0] | Pin 1 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 11:8 | SEL10[3:0] | Pin 10 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 7:4 | SEL9[3:0] | Pin 9 alternate function selected<br>These bits are set and cleared by software.<br>Refer to SEL8[3:0] description |
| 3:0 | SEL8[3:0] | Pin 8 alternate function selected<br>These bits are set and cleared by software.<br>0000: AF0 selected (reset value)<br>0001: AF1 selected<br>0010: AF2 selected<br>0011: AF3 selected<br>0100: AF4 selected<br>0101: AF5 selected<br>0110: AF6 selected<br>0111: AF7 selected<br>1000: AF8 selected<br>1001: AF9 selected<br>1010: AF10 selected<br>1011: AF11 selected<br>1100: AF12 selected<br>1101: AF13 selected<br>1110: AF14 selected<br>1111: AF15 selected |

### 10.4.11. Bit clear register (GPIOx_BC, x=A…H, J, K)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CR15 | CR14 | CR13 | CR12 | CR11 | CR10 | CR9 | CR8 | CR7 | CR6 | CR5 | CR4 | CR3 | CR2 | CR1 | CR0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CRy | Port clear bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Clear the corresponding OCTLy bit |

## 10.4.12. Port bit toggle register (GPIOx_TG, x=A…H, J, K)

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TG15 | TG14 | TG13 | TG12 | TG11 | TG10 | TG9 | TG8 | TG7 | TG6 | TG5 | TG4 | TG3 | TG2 | TG1 | TG0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

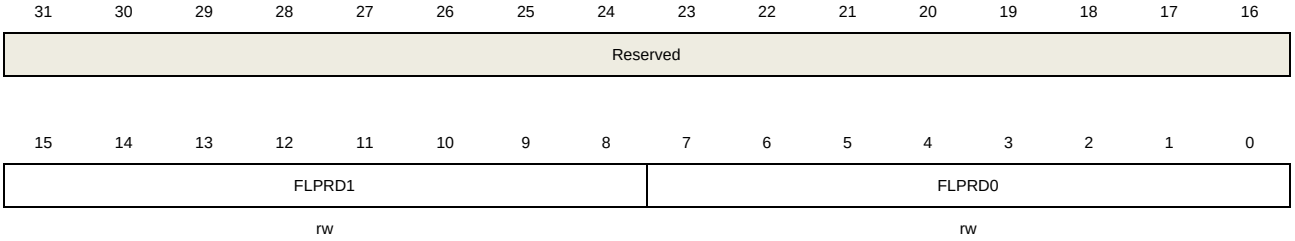
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | TGy | Port toggle bit y(y=0..15)<br>These bits are set and cleared by software.<br>0: No action on the corresponding OCTLy bit<br>1: Toggle the corresponding OCTLy bit |

## 10.4.13. Input filtering register (GPIOx_IFL, x=A...H, J, K)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).



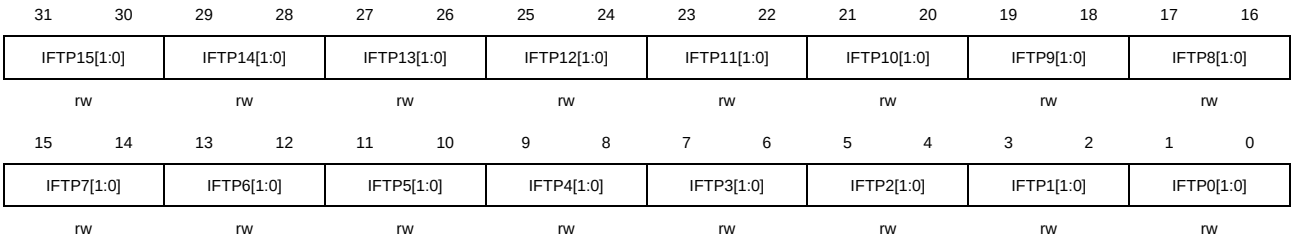

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:8 | FLPRD1 | Filter sampling frequency for GPIO8 to GPIO15:<br>00: FLPRDx = CK_AHB<br>01: FLPRDx = CK_AHB / 2<br>02: FLPRDx = CK_AHB / 4<br>....<br>FF: FLPRDx = CK_AHB / 510 |
| 7:0 | FLPRD0 | Filter sampling frequency for GPIO1 to GPIO7:<br>00: FLPRDx = CK_AHB<br>01: FLPRDx = CK_AHB / 2<br>02: FLPRDx = CK_AHB / 4<br>....<br>FF: FLPRDx = CK_AHB / 510 |

## 10.4.14. Input filtering type register (GPIOx_IFTP, x=A...H, J, K)

Address offset: 0x34

Reset value: 0xFFFF FFFF

This register can be write by byte (8-bit), half-word (16-bit) and word (32-bit) and be read by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | IFTP15[1:0] | Pin 15 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 29:28 | IFTP14[1:0] | Pin 14 input filtering type bits |

| | | |
|---|---|---|
| | | These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 27:26 | IFTP13[1:0] | Pin 13 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 25:24 | IFTP12[1:0] | Pin 12 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 23:22 | IFTP11[1:0] | Pin 11 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 21:20 | IFTP10[1:0] | Pin 10 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 19:18 | IFTP9[1:0] | Pin 9 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 17:16 | IFTP8[1:0] | Pin 8 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 15:14 | IFTP7[1:0] | Pin 7 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 13:12 | IFTP6[1:0] | Pin 6 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 11:10 | IFTP5[1:0] | Pin 5 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 9:8 | IFTP4[1:0] | Pin 4 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 7:6 | IFTP3[1:0] | Pin 3 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 5:4 | IFTP2[1:0] | Pin 2 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |

| | | |
|---|---|---|
| 3:2 | IFTP1[1:0] | Pin 1 input filtering type bits<br>These bits are set and cleared by software.<br>Refer to IFTP0[1:0] description |
| 1:0 | IFTP0[1:0] | Pin 0 input filtering type bits<br>These bits are set and cleared by software.<br>00: Synchronization<br>01: Filtering (3 samples)<br>10: Filtering (6 samples)<br>11: Asynchronous (no synchronization or filtering) |

# 11. Cyclic redundancy checks management unit (CRC)

## 11.1. Overview

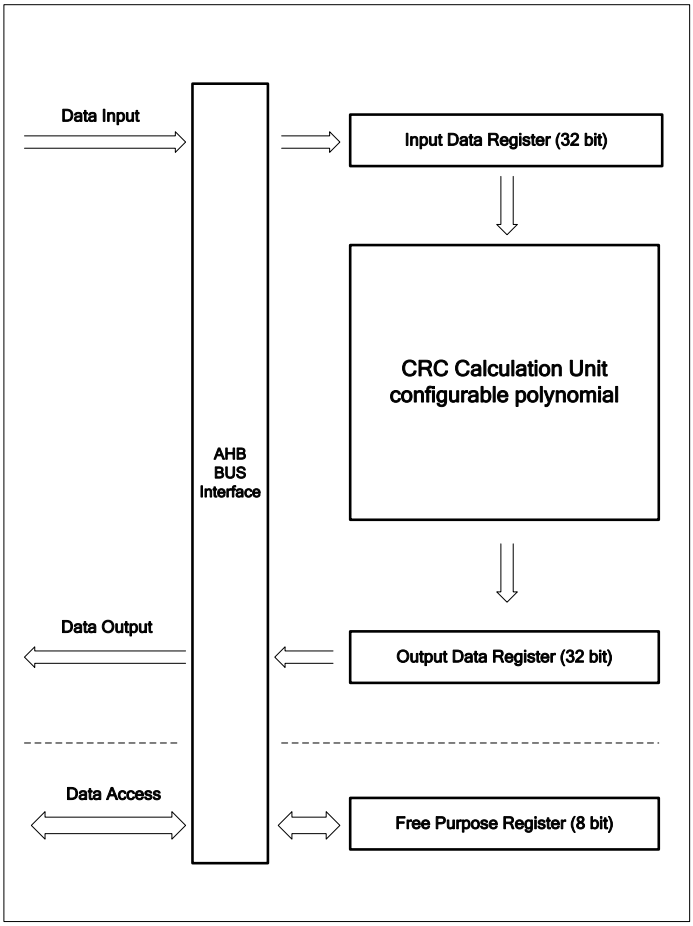
A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

This CRC management unit can be used to calculate 7 / 8 / 16 / 32 bits CRC code within user configurable polynomial.

## 11.2. Characteristics

- Supports 7 / 8 / 16 / 32 bits data input.
- For 7(8) / 16 / 32 bits input data length, the calculation cycles are 1 / 2 / 4 AHB clock cycles.
- User configurable polynomial value and size.
- After CRC module·reset, user can configure initial value.
- Free 8-bit register is unrelated to calculation and can be used for any other goals by any other peripheral devices.

**Figure 11-1. Block diagram of CRC calculation unit**




## 11.3. Function overview

- CRC calculation unit is used to calculate the 32-bit raw data, and CRC_DATA register will receive the raw data and store the calculation result.

  If the CRC_DATA register has not been cleared by setting the CRC_CTL register, the new input raw data will be calculated based on the result of previous value of CRC_DATA.

  CRC calculation will spend 4 / 2 / 1 AHB clock cycles for 32 / 16 / 8(7) bits data size. During this period, AHB will not be hanged because of the existence of the 32-bit input buffer.

- This module supplies an 8-bit free register CRC_FDATA.

  CRC_FDATA is unrelated to the CRC calculation. Independent read and write operations can be performed at any time.

- Reversible function can reverse the input data and output data.

  For input data, 3 reverse types can be selected.

  Original data is 0x3456CDEF:

1) byte reverse:
32-bit data is divided into 4 groups and reverse implement in group inside. Reversed data: 0x2C6AB3F7

2) half-word reverse:
32-bit data is divided into 2 groups and reverse implement in group inside. Reversed data: 0x6A2CF7B3

3) word reverse:
32-bit data is divided into 1 group and reverse implement in group inside. Reversed data: 0xF7B36A2C

For output data, reverse type is word reverse.

For example: when REV_O=1, calculation result 0x3344CCDD will be converted to 0xBB3322CC.

- User configurable initial calculation data is available.

  When RST bit is set or write operation to CRC_IDATA register, the CRC_DATA register will be automatically initialized to the value in CRC_IDATA.

- User configurable polynomial.

  Depends on PS[1:0] bits, the valid polynomial and output bit width can be selected by user. If the polynomial is less than 32-bit, the high bits of the input data and output data is unavailable. It is strongly recommend resetting the CRC calculation unit after change the PS[1:0] bits or polynomial.

## 11.4. Register definition

CRC base address: 0x4002 3000

### 11.4.1. Data register (CRC_DATA)

Address offset: 0x00
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DATA[31:0] | CRC calculation result bits<br>Software writes and reads.<br>This register is used to calculate new data, and the register can be written the new data directly. Write value cannot be read because the read value is the previous CRC calculation result. |

### 11.4.2. Free data register (CRC_FDATA)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | FDATA[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | FDATA[7:0] | Free data register bits<br>Software writes and reads.<br>These bits are unrelated with CRC calculation. This byte can be used for any goal by any other peripheral. The CRC_CTL register will generate no effect to the byte. |

### 11.4.3. Control register (CRC_CTL)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | REV_O | REV_I[1:0] | | PS[1:0] | | Reserved | | RST |
| | | | | | | | | rw | rw | | rw | | | | rs |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | REV_O | Reverse output data value in bit order<br>0:Not bit reversed for output data<br>1:Bit reversed for output data |
| 6:5 | REV_I[1:0] | Reverse type for input data<br>0: Dot not use reverse for input data<br>1: Reverse input data with every 8-bit length<br>2: Reverse input data with every 16-bit length<br>3: Reverse input data with whole 32-bit length |
| 4:3 | PS[1:0] | Size of polynomial<br>0: 32-bit<br>1: 16-bit ( POLY [15:0] is used for calculation. )<br>2: 8-bit ( POLY [7:0] is used for calculation. )<br>3: 7-bit ( POLY [6:0] is used for calculation. ) |
| 2:1 | Reserved | Must be kept at reset value. |
| 0 | RST | Software writes and reads.<br>Set this bit can reset the CRC_DATA register.<br>When set, the value of the CRC_DATA register is automatically initialized to the value in the CRC_IDATA register and then automatically cleared by hardware. This bit will take no effect to CRC_FDATA. |

### 11.4.4. Initialization data register (CRC_IDATA)

Address offset: 0x10

Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| IDATA[31:16] | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IDATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | IDATA[31:0] | Configurable initial CRC data value<br>When RST bit in CRC_CTL asserted, CRC_DATA will be programmed to this value. |

### 11.4.5. Polynomial register (CRC_POLY)

Address offset: 0x14

Reset value: 0x04C1 1DB7

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POLY[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| POLY[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | POLY[31:0] | User configurable polynomial value<br>This value is used together with PS[1:0] bits. |

# 12. True random number generator (TRNG)

## 12.1. Overview

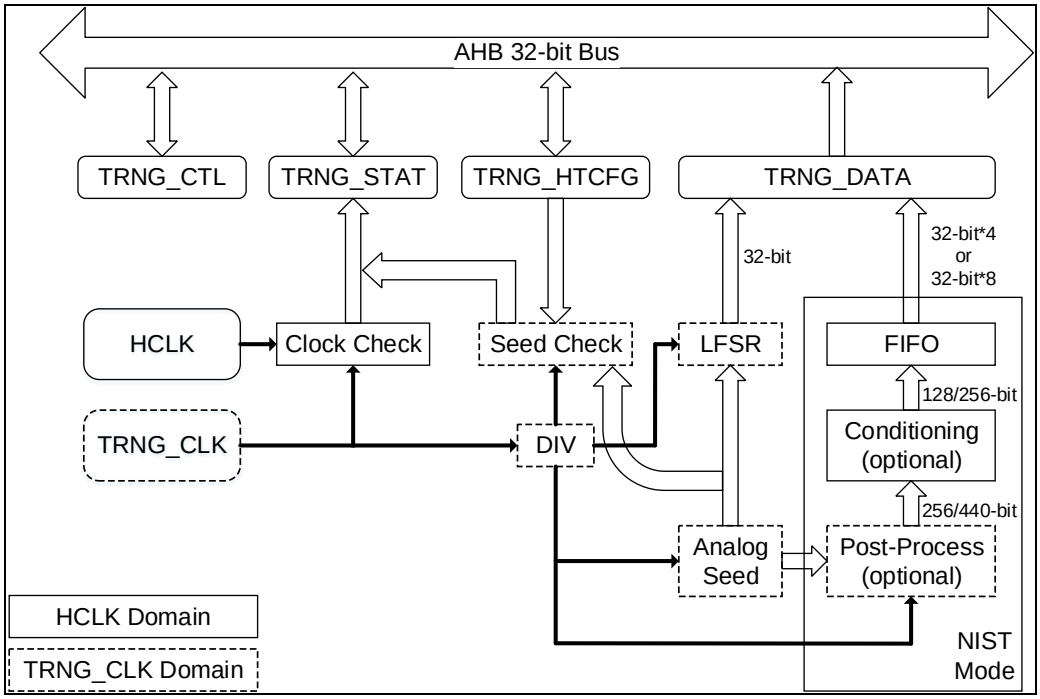
The true random number generator (TRNG) module can generate a 32-bit random value by using continuous analog noise and it has been pre-certified NIST SP800-90B.

## 12.2. Characteristics

- LFSR (Linear Feedback Shift Register) mode and NIST (National Institute of Standards and Technology) mode to generate random number.
- About 40 periods of TRNG_CLK are needed between two consecutive random numbers in LFSR mode.
- 32-bit random numbers are generated each time in LFSR mode.
- TRNG NIST mode follows the NIST SP800-90B.
- Support health tests recommended by the NIST SP800-90B.
- 4 * 32-bit or 8 * 32-bit random numbers are generated each time in NIST mode.
- TRNG has the functions of startup and in-service self-check, associated with specific error flags.
- Disable TRNG module will reduce the chip power consumption.
- 128-bit random value seed is generated from analog noise, so the random number is a true random number.

## 12.3. Function overview

**Figure 12-1. TRNG block diagram**




There are two modes in TRNG module, NIST mode and LFSR mode.

In the NIST mode, the analog seeds of random number come from 4 noise sources. The noise source signal is first operated by XOR and then digitized to obtain a 1-bit analog seed. This analog seed is then plugged into a conditioning logic (uint) to generate 128-bit or 256-bit data output. A HASH function complied with NIST SP800-90B is implemented in the conditioning stage to increase the quality of random number. And the output data can be read out by reading the data register TRNG_DATA 4 or 8 times continuously after the data ready flag (DRDY) in TRNG_STAT register is set.

In the LFSR mode, the analog random number seed is plugged into a linear feedback shift register (LFSR), where a 32-bit width random number is generated. The 32-bit value of LFSR will transfer into TRNG_DATA register after a sufficient number of seeds have been sent to the LFSR.

The analog seed is generated by several ring oscillators. The seed generate model is driven by a configurable TRNG_CLK (refer to RCU chapter), so that the quality of the generated random number depends on TRNG_CLK exclusively, no matter what HCLK frequency was.

### 12.3.1. LFSR

A Linear Feedback Shift Register is a sequential shift register with combinational logic that causes it to pseudo-randomly cycle through a sequence of binary values. This operation will increase the quality of the random number, and TRNG generates 32-bit data each operation in this mode.

### 12.3.2. Post processing

When this function is enabled, half of the bits are taken from the sampled noise source, half of the bits are taken from inverted sampled noise source. And when the output data is ready, the post-process module will start a new random seed collection even if the random data haven't been read out. This operation will increase efficiency of TRNG module.

### 12.3.3. Conditioning

The conditioning component in the TRNG is a deterministic HASH function that increases the quality of the resulting fixed-length bitstrings output. The NIST SP800-90B target is full entropy on the output.

The hash operation is controlled by ALGO[1:0] bits field and INIT bit in TRNG_CTL register. And the input random number bit width is controlled by register INMOD (256-bit or 440-bit), while the output bit width is controlled by OUTMOD (128-bit or 256-bit).

### 12.3.4. Output FIFO

The width of data output FIFO is 4 or 8 (controlled by OUTMOD) words(32-bit) in NIST mode, and these data are stored in the output FIFO temporarily. When all words have been read from the output FIFO through the TRNG_DATA register, a new round of conditioning process is automatically started, then another 128-bit or 256-bit conditioning output data is pushed into the output FIFO and waiting for the next reading.

The DRDY bit in TRNG_STAT register will be set when a random number is available through the TRNG_DATA register. This bit remains set until output FIFO becomes empty after reading 4 or 8 times continuously from the TRNG_DATA register in NIST mode. This flag also remains setting until the one TRNG_DATA is read out when in LFSR mode.

In NIST mode, it will cost about input seed number(controlled by INMOD bit in the TRNG_CTL register) plus 10 TRNG_CLK periods and 70 HCLK periods. While in LFSR mode, generate a 32bit random number need about 40 TRNG_CLK periods.

The hash function with different ALGO setting in the TRNG_CTL register will produce results of different valid length. TRNG module will place the valid bits at high order of output FIFO, so the first read data is always valid data. And user should select the expect data and register setting according to the need. More details refer to ***Table 12-1. ALGO configurations***.

**Table 12-1. ALGO configurations**

| ALGO | 00 | 01 | 10 | 11 |
|---|---|---|---|---|
| **algorithm** | SHA1 | MD5 | SHA224 | SHA256 |
| **valid length** | 160 | 128 | 224 | 256 |

### 12.3.5. Health tests

This component ensures the stable operation of TRNG and can quickly monitor the

occurrence of errors.

The health tests features of TRNG module following NIST SP800-90B. For more details about thresholds, refer to TRNG_HTCFG register.

1) Start-up health tests: These tests performed after reset and before using TRNG to get random numbers for the first time.
   - Adaptive proportion test: The TRNG verifies that the first bit on the outputs of the noise source is not repeated more than a threshold value, this threshold value refer to APT_TH bit fields in the TRNG_HTCFG register (default 691 times).
   - Repetition count test: If the noise source has provided more than a specified number of consecutive bits at a constant value(0 or 1), an error will occur and the relevant error flag will be set. This threshold value refer to RCT_TH bit fields in TRNG_HTCFG register(default 40).
   - Replace test: The TRNG can replace the input of condition stage with a specific number, and compare the output to the correct answer according to different algorithm during this stage.
2) Continuous health tests: These tests are run indefinitely on the outputs of the noise source while the noise source is operating.
   - Adaptive proportion test: refer to the start-up health tests.
   - Repetition count test: refer to the start-up health tests.
3) GD-Defined continuous health tests: Additional health tests specified by vendor.
   - Transition count test: if the noise source provided more than 32 consecutive occurrence of two bits patterns(01 or 10), an error will occur and the relevant error flag will be set.
   - Clock detector: if the TRNG clock cycle before divided is smaller than AHB clock cycle divided by 16, an error will occur and the relevant error flag will be set.
4) On-demand test of the noise source output
   - The noise source output only support on-demand testing by restarting the entropy source and rerunning the startup tests.

**Note:**

- In NIST mode, ERR_STA bit in the TRNG_STAT register will be set when an error is detected, and if the interrupt bit of TRNG is asserted, an interrupt is generated.
- When the replace test is enabled, the random numbers generated are only used to verify the functionality of the conditioning component. After the test is completed, the random numbers should be discarded and should not be used as true random numbers.

### 12.3.6. NIST mode state

The states of NIST mode are shown below:

1. The initial state of the TRNG is idle state.
2. It goes to warm-up state after enabling the TRNG by setting RNGEN bit in the TRNG_CTL register. This state is a period for analog initialition.
3. After counting 16 cycles of TRNG_CLK (before divider), the state goes to start-up state, and the start-up health tests are implemented, which will cost 1024 divided TRNG_CLK cycles.
4. Then the state changes to gen-samp (generate sample) state to generate random number sample. The TRNG module will generate a new set of random number after the output FIFO is empty.

If an error occurs when the TRNG is in start-up state or gen-samp state, the ERR_STA bit in the TRNG_STAT register will be set.

**Note :**

- Don't change the CLKDIV[3:0] bits field in TRNG_CTL register when in operation.
- When the TRNG is in start-up state, a random number is also generated for the first sample generate time reduce consideration, even if this sample is suggested to discard according to the FIPS PUB140-2.
- If RT_EN bit in the TRNG_CTL register is set, the first output number is a replace test result, and it should be discarded.

### 12.3.7. Operation flow

The following steps are recommended for using TRNG block:

1. Set CONDRST bit in TRNG_CTL register.
2. Write the required configuration in the TRNG_CTL register, such as module power consumption, clock frequency division factor, operating mode, input / output bit-width, algorithm, etc.
3. Enable the TRNGEN bit.
4. Clear the CONDRST bit for the written configuration to take effect.
5. Check the status register TRGN_STAT, if SEIF, CEIF, ERR_STA, SECS and CECS are all stay 0 and DRDY=1, then the random value in the data register could be read.

When the IE bit in the TRNG_CTL register is set, an interrupt is generated if:

- Successfully generated a random number, and the DRDY bit in the TRNG_STAT register is set.
- A seed error occurs, and the SEIF and ERR_STA bits in the TRNG_STAT register are set.
- A clock error occurs, and the CEIF and ERR_STA bits in the TRNG_STAT register are set.

As required by the FIPS PUB 140-2, the first random data in data register should be saved but not be used. Every subsequent new random data should be compared to the previously random data. The data can only be used if it is not equal to the previously one.

### 12.3.8. Error flags

(1) Clock error

When the TRNG_CLK frequency is lower than the 1/16 of HCLK, the CECS and CEIF bit will be set. In this case, the application should check TRNG_CLK and HCLK frequency configurations and then clear CEIF bit. Clock error will not impact the previous random data.

(2) Seed error

When the analog seed is not changed or always changing exceed the threshold, the SECS and SEIF bit will be set. In this case, the random data in data register should not be used. The application needs a TRNG software reset by writing CONDRST bit at 1 and then at 0, clear the SEIF bit after reset TRNG, then wait for SECS bit in the TRNG_STAT register to be cleared by TRNG.

### 12.3.9. Low power usage

If customers concern about power consumption, besides configure the CLKDIV a large number, they can also disable the TRNG module after setting the DRDY bit in TRNG_STAT register. TRNG module will remain at the last state, and the random data still can be read through the register TRNG_DATA. If a new random number is required, enable TRNG to activate the random number generation again.

## 12.4. Register definition

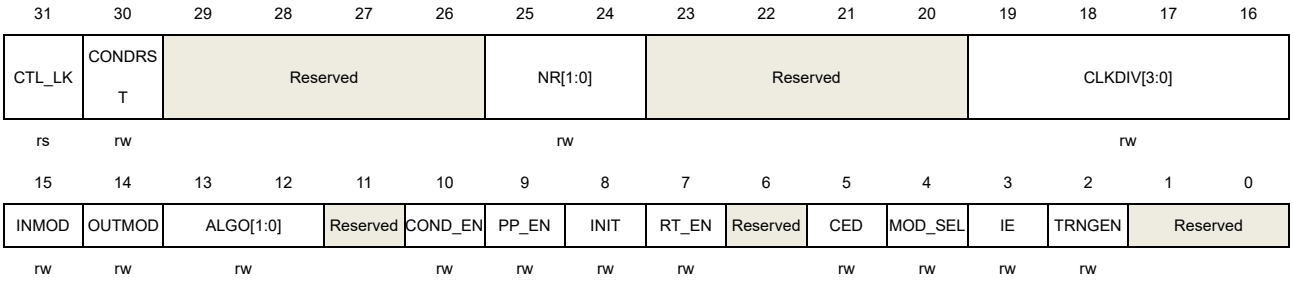
TRNG base address: 0x4802 1800

### 12.4.1. Control register (TRNG_CTL)

Address offset: 0x00

Reset value: 0x0300 0410

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CTL_LK | TRNG_CTL register lock bit<br>This bit can only be reset to 0 if TRNG is reset.<br>0: Write bits[29:4] are allowed<br>1: Lock bits[29:4] and write bits[29:4] are ignored |
| 30 | CONDRST | Reset conditioning logic<br>Write 1 then write 0 to reset conditioning logic. It should be noted that TRNG_HTCFG register and bits[29:4] in the TRNG_CTL register can only be written when CONDRST is 1.<br>Note: During the period from 1 to 0 of this bit, at least three TRNG_CLK clock signals after frequency division need to be maintained to ensure the synchronization of configuration data |
| 29:26 | Reserved | Must be kept at reset value. |
| 25:24 | NR[1:0] | Analog trng power mode. Reset value:2'b 11<br>00: Ultra low;<br>01: Low<br>10: Medium<br>11: High |
| 23:20 | Reserved | Must be kept at reset value. |
| 19:16 | CLKDIV[3:0] | TRNG clock divider<br>0000: $2^0$ TRNG clock cycle per internal TRNG clock<br>0001: $2^1$ TRNG clock cycles per internal TRNG clock<br>…… |

| | | |
|---|---|---|
| | | 1111: $2^{15}$ TRNG clock cycles per internal TRNG clock |
| 15 | INMOD | Select random seed number input to conditioning module<br>0: 256 bits<br>1: 440 bits |
| 14 | OUTMOD | Select random data width output of conditioning module<br>0: 128-bit<br>1: 256-bit |
| 13:12 | ALGO[1:0] | conditioning module hash algorithm selection<br>00: SHA1<br>01: MD5<br>10: SHA224<br>11: SHA256 |
| 11 | Reserved | Must be kept at reset value. |
| 10 | COND_EN | The enable bit of conditioning component<br>0: Disable conditioning component<br>1: Enable conditioning component |
| 9 | PP_EN | The enable bit of post_processing function<br>0: Disable post_processing function<br>1: Enable post_processing function |
| 8 | INIT | Initialize hash algorithm when conditioning enabled.<br>0: Deinitialize hash algorithm<br>1: Initialize hash algo algorithm |
| 7 | RT_EN | Replace test enable bit<br>0: Disable replace test<br>1: Enable replace test |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CED | Clock error detection<br>0: Disable clock error detection<br>1: Enable clock error detection |
| 4 | MOD_SEL | LFSR or NIST mode selection<br>0: LFSR mode<br>1: NIST mode |
| 3 | IE | The enable bit of the TRNG interrupt. This bit controls the generation of an interrupt when DRDY, SEIF, CEIF or ERR_STA was set.<br>0: Disable TRNG interrupt<br>1: Enable TRNG interrupt |
| 2 | TRNGEN | The enabled bit of the TRNG.<br>0: Disable TRNG module (reduce power consuming) |

| | | |
|---|---|---|
| | | 1: Enable TRNG module |
| 1:0 | Reserved | Must be kept at reset value. |

### 12.4.2. Status register (TRNG_STAT)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | SEIF | CEIF | Reserved | ERR_STA | SECS | CECS | DRDY |
| | | | | | | | | | rc_w0 | rc_w0 | | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | SEIF | Seed error interrupt flag<br>This bit will be set if more than 64 consecutive same bit or more than 32 consecutive 01(or 10) changing or adaptive proportion test window exceeds APT_TH repetition during operation are detected.<br>0: No fault detected<br>1: Seed error has been detected. The bit is cleared by writing 0. |
| 5 | CEIF | Clock error interrupt flag<br>This bit will be set if TRNG_CLK frequency is lower than 1/16 HCLK frequency.<br>0: No fault detected<br>1: Clock error has been detected. The bit is cleared by writing 0. |
| 4 | Reserved | Must be kept at reset value. |
| 3 | ERR_STA | NIST mode error flag, this bit could be reset by CONDRST<br>0: No error occurs in NIST mode<br>1: Error occurs in NIST mode |
| 2 | SECS | Seed error current status<br>0: Seed error is not detected at current time. In case of SEIF=1 and SECS=0, it means seed error has been detected before but now is recovered.<br>1: Seed error is detected at current time if more than 64 consecutive same bits or more than 32 consecutive 01(or 10) changing or adaptive proportion test window exceeds APT_TH repetition during operation are detected |
| 1 | CECS | Clock error current status<br>0: Clock error is not detected at current time. In case of CEIF=1 and CECS=0, it means clock error has been detected before but now is recovered. |

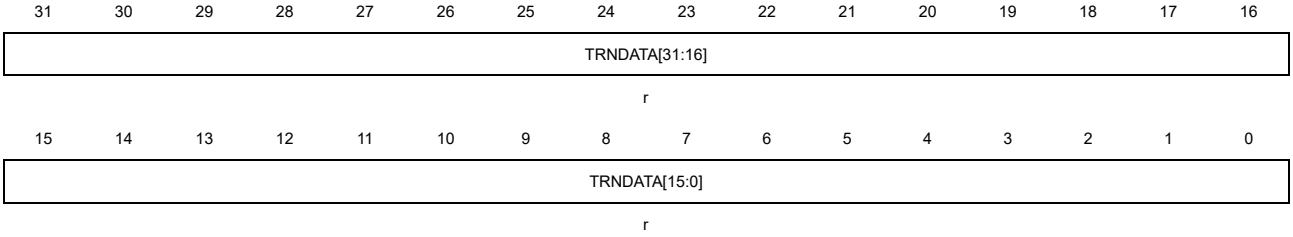
| | | |
|---|---|---|
| | | 1: Clock error is detected at current time. TRNG_CLK frequency is lower than 1/16 HCLK frequency. |
| 0 | DRDY | Random data ready status bit. This bit is cleared by reading the TRNG_DATA register and set when a new random number is generated.<br>0: The content of TRNG data register is not available.<br>1: The content of TRNG data register is available. |

### 12.4.3. Data register (TRNG_DATA)

Address offset: 0x08
Reset value: 0x0000 0000

Application must make sure DRDY bit in the TRNG_STAT register is set before reading this register.

This register has to be accessed by word (32-bit).



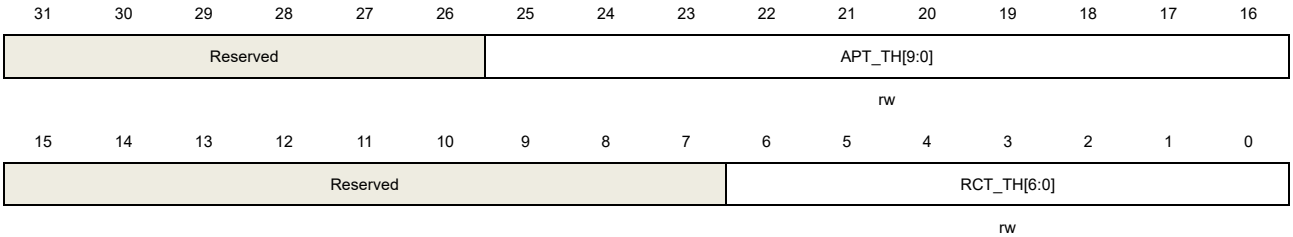

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TRNDATA[31:0] | 32-bit random data |

### 12.4.4. Health tests configure register (TRNG_HTCFG)

Address offset: 0x10
Reset value: 0x02B3 0028

Set CONDRST bit and clear CTL_LK bit in the TRNG_CTL before writing TRNG_HTCFG.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25:16 | APT_TH[9:0] | Adaptive proportion test threshold. Default 691. |

| | | |
|---|---|---|
| 15:7 | Reserved | Must be kept at reset value. |
| 6:0 | RCT_TH[6:0] | Repetition count test threshold. Default 40. |

# 13. Cryptographic Acceleration Unit (CAU) (For GD32H737_757_759)

## 13.1. Overview

The cryptographic acceleration unit (CAU) is used to encipher and decipher data with DES, Triple-DES or AES (128, 192, or 256) algorithms. This module follows the following standards:

- The Data Encryption Standard (DES) and the Triple Data Encryption Algorithm (TDEA) are announced by Federal Information Processing Standards Publication (FIPS) 46-3, October 25, 1999. It follows the American National Standards Institute (ANSI) X9.52 standard.
- The Advanced Encryption Standard (AES) is announced by Federal Information Processing Standards Publication 197, November 26, 2001.

DES / TDES / AES algorithms with different key sizes are supported to perform data encryption and decryption in the CAU in multiple modes.

The CAU is a 32-bit peripheral, DMA transfer is supported and data can be accessed in the input and output FIFO.

## 13.2. Characteristics

- DES, TDES and AES encryption / decryption algorithms are supported.
- Multiple modes are supported respectively in DES, TDES and AES, including Electronic codebook (ECB), Cipher block chaining (CBC), Counter mode (CTR), Galois / counter mode (GCM), Galois message authentication code mode (GMAC), Counter with CBC-MAC (CCM), Cipher Feedback mode (CFB) and Output Feedback mode(OFB).
- DMA transfer for incoming and outgoing data is supported.

DES / TDES

- Supports the ECB and CBC chaining algorithms.
- Two 32-bit initialization vectors (IV) are used in CBC mode.
- 8*32-bit input and output FIFO.
- Multiple data types are supported, including No swapping, Half-word swapping Byte swapping and Bit swapping.
- Data are transferred by DMA, CPU during interrupts, or without both of them.

AES

- Supports the ECB, CBC, CTR, GCM, GMAC, CCM, CFB and OFB chaining algorithms.
- Supports 128-bit, 192-bit and 256-bit keys.
- Four 32-bit initialization vectors (IV) are used in CBC, CTR, GCM, GMAC, CCM, CFB and OFB modes.

- 8*32-bit input and output FIFO.
- Multiple data types are supported, including No swapping, Half-word swapping Byte swapping and Bit swapping.
- Data can be transferred by DMA, CPU during interrupts, or without both of them.
- Support to use initial key from EFUSE.

## 13.3. CAU data type and initialization vectors

### 13.3.1. Data type

The cryptographic acceleration unit receives data of 32 bits at a time, while they are processed in 64 / 128 bits for DES / AES algorithms. For each data block, according to the data type, the data could be bit / byte / half-word / no swapped before they are transferred into the cryptographic acceleration processor. The same swapping operation should be also performed on the processor output data before they are collected. Note the least-significant data always occupies the lowest address location no matter which data type is configured, because the system memory is little-endian.

*Figure 13-1. DATAM No swapping and Half-word swapping* and *Figure 13-2. DATAM Byte swapping and Bit swapping* illustrate the 128-bit AES block data swapping according to different data types. (For DES, the data block is two 32-bit words, please refer to the first two words data swapping in the figure).

**Figure 13-1. DATAM No swapping and Half-word swapping**

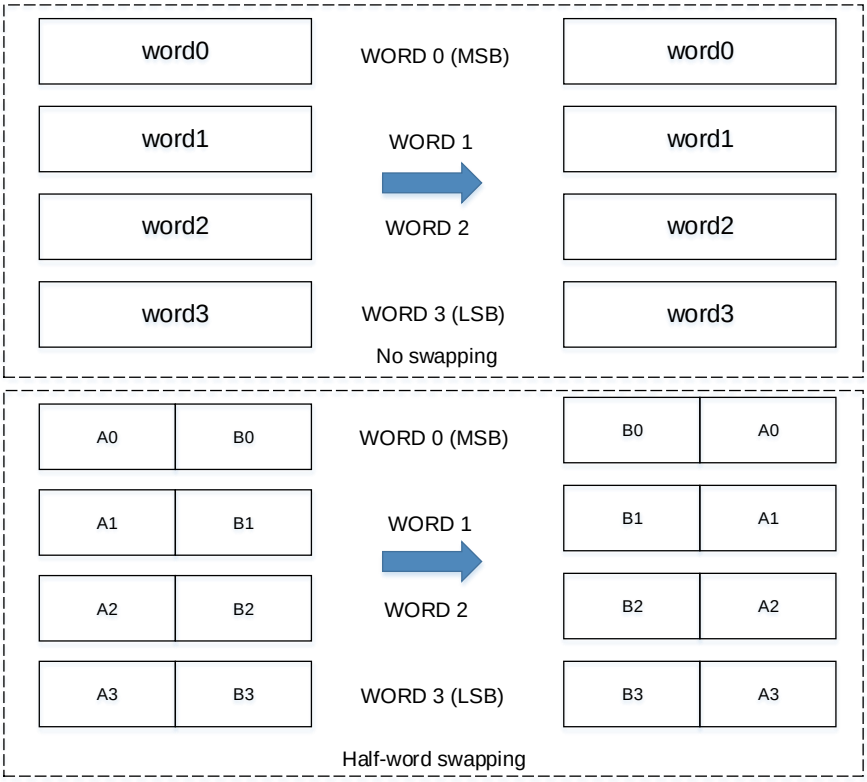
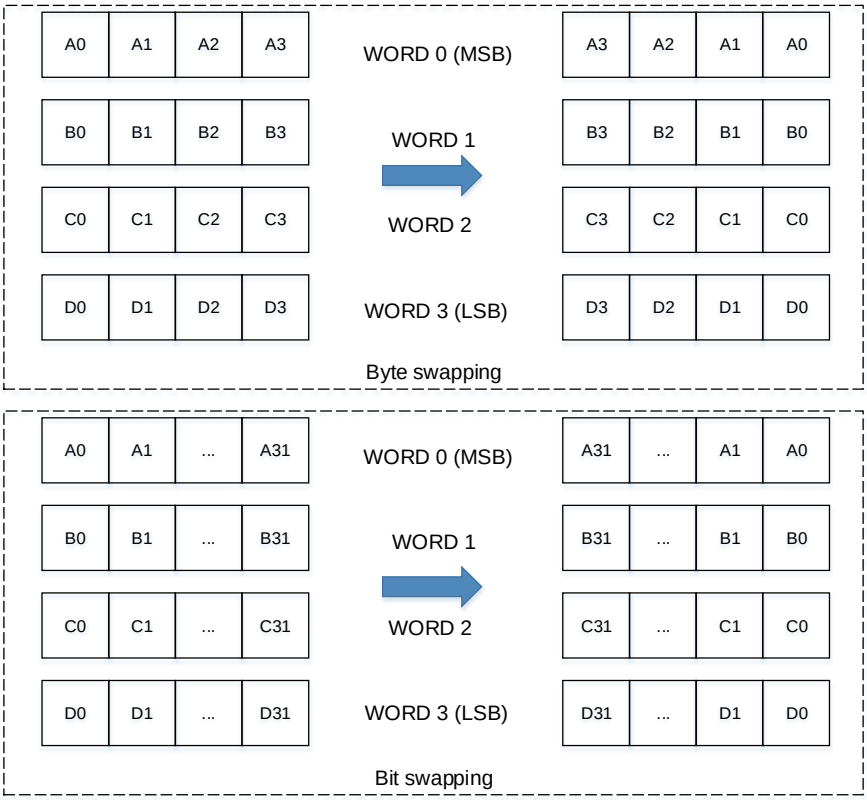


**Figure 13-2. DATAM Byte swapping and Bit swapping**




### 13.3.2. Initialization vectors

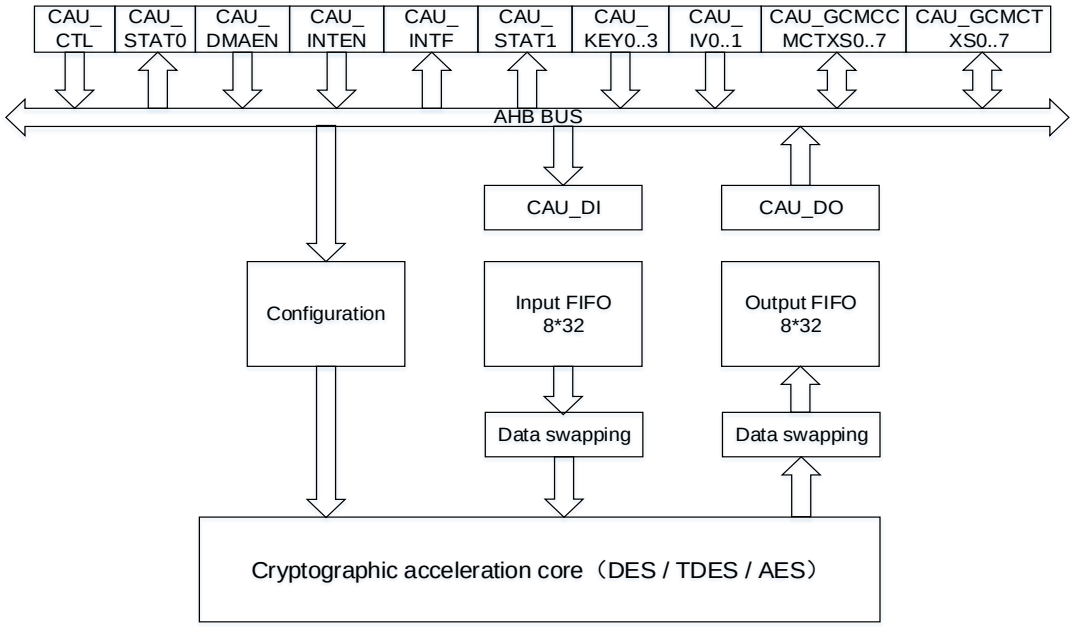
The initialization vectors are used in CBC, CTR, GCM, GMAC, CCM, CFB and OFB modes to XOR with data blocks. They are independent of plaintext and ciphertext, and the DATAM value will not affect them. Note the initialization vector registers CAU_IV0..1(H/L) can only be written when BUSY is 0, otherwise the write operations are invalid.

## 13.4. Cryptographic acceleration processor

The cryptographic acceleration unit implements DES and AES acceleration processors, which are detailed described in section ***DES / TDES cryptographic acceleration*** processor and ***AES cryptographic acceleration processor***.

***Figure 13-3. CAU diagram*** shows the block diagram of the cryptographic acceleration unit.

**Figure 13-3. CAU diagram**




### 13.4.1. DES / TDES cryptographic acceleration processor

The DES/TDES cryptographic acceleration processor contains the DES algorithm (DEA), cryptographic keys (1 for DES algorithm and 3 for TDES algorithm), and initialization vectors in CBC mode.

#### DES / TDES key

[KEY1] is used in DES and [KEY3 KEY2 KEY1] are used in TDES respectively.
When TDES algorithm is configured, three different keying options are allowed:

1. Three same keys

The three keys KEY3, KEY2 and KEY1 are completely equal, which means KEY3=KEY2=KEY1. FIPS PUB 46-3 – 1999 (and ANSI X9.52 -1998) refers to this option. It is easy to understand that this mode is equivalent to DES.

2. Two different keys

In this option, KEY2 is different from KEY1, and KEY3 is equal to KEY1, which means, KEY1 and KEY2 are independent while KEY3= KEY1. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option.

3. Three different keys

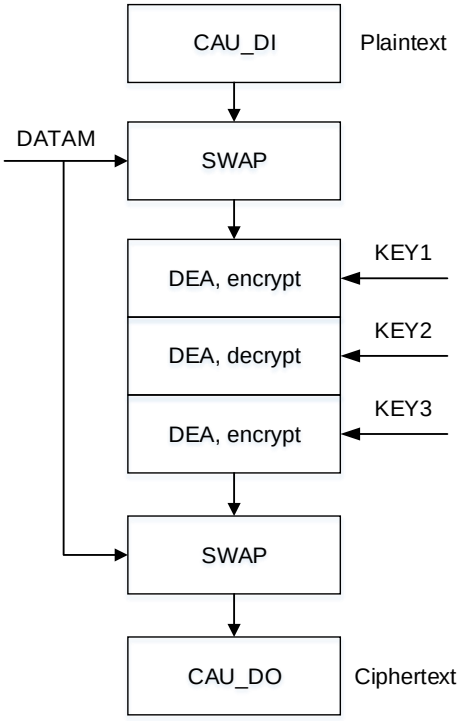
In this option, KEY1, KEY2 and KEY3 are completely independent. FIPS PUB 46-3 -1999 (and ANSI X9.52 – 1998) refers to this option.

More information of the thorough explanation of the key used in the DES/TDES please refer to FIPS PUB 46-3 (and ANSI X9.52 -1998), and the explanation process is omitted in this manual.

### DES / TDES ECB encryption

The 64-bit input plaintext is first obtained after data swapping according to the data type. When the TDES algorithm is configured, the input data block is read in the DEA and encrypted using KEY1. The output is fed back directly to next DEA and then decrypted using KEY2. After that, the output is fed back directly to the last DEA and encrypted with KEY3. The output after above processes is then swapped back according to the data type again, and a 64-bit ciphertext is produced. When the DES algorithm is configured, the result of the first DEA encrypted using KEY1 is swapped directly according to the data type, and a 64-bit ciphertext is produced. The procedure of DES / TDES ECB mode encryption is illustrated in ***Figure 13-4. DES / TDES ECB encryption***.

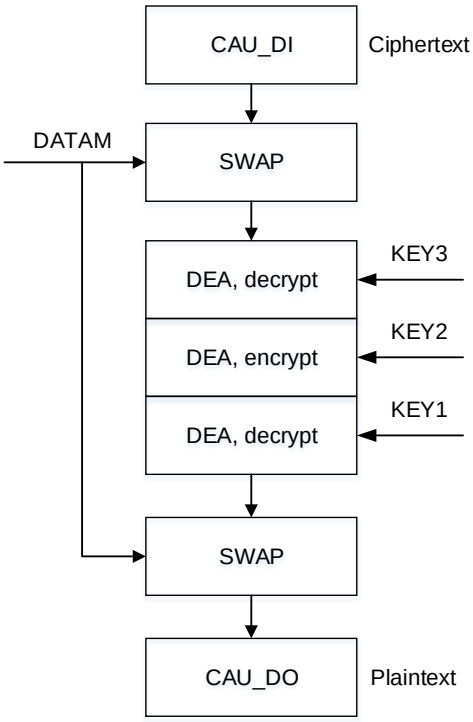
**Figure 13-4. DES / TDES ECB encryption**




### DES/TDES ECB decryption

The 64-bit input ciphertext is first obtained after data swapping according to the data type. When the TDES algorithm is configured, the input data block is read in the DEA and decrypted using KEY3. The output is fed back directly to next DEA and then encrypted using KEY2. After that, the output is fed back directly to the last DEA and decrypted with KEY1. The output after above process is then swapped back according to the data type again, and a 64-bit plaintext is produced. When the DES algorithm is configured, the result of the first DEA decrypted using KEY1 is swapped directly according to the data type, and a 64-bit plaintext is produced. The procedure of DES/TDES ECB mode decryption is illustrated in ***Figure 13-5. DES / TDES ECB decryption***.

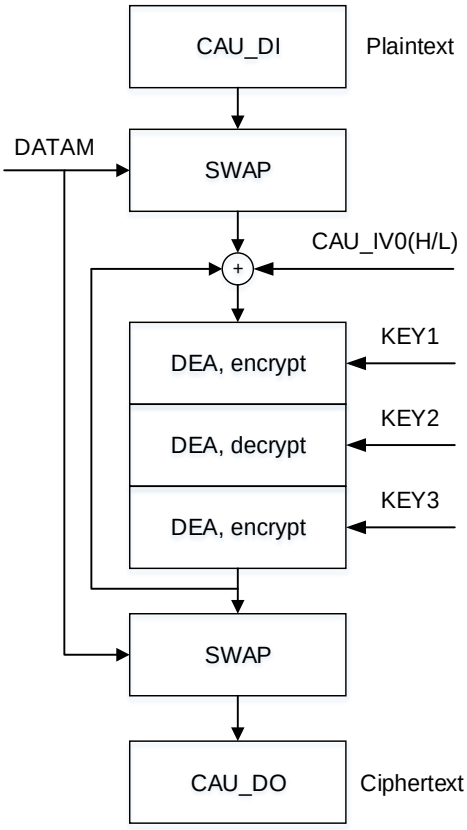
**Figure 13-5. DES / TDES ECB decryption**




### DES / TDES CBC encryption

The input data of the DEA block in CBC mode consists of two aspects: the input plaintext after data swapping according to the data type, and the initialization vectors. When the TDES algorithm is configured, the XOR result of the swapped plaintext data block and the 64-bit initialization vector CAU_IV0..1 is read in the DEA and encrypted using KEY1. The output is fed back directly to next DEA and then decrypted using KEY2. After that, the output is fed back directly to the last DEA and encrypted with KEY3. The result is then used as the next initialization vector and exclusive-ORed with the next plaintext data block to process next encryption. The above operations are repeated until the last plaintext block is encrypted. Note that the final partial data block should be encrypted in a specified manner if the plaintext message does not consist of an integral number of data blocks. At last, the output ciphertext is also obtained after data swapping according to the data type. When the DES algorithm is configured, the state and process of the second and third block of DEA should be omitted. The procedure of DES / TDES CBC mode encryption is illustrated in ***Figure 13-6. DES / TDES CBC encryption***.

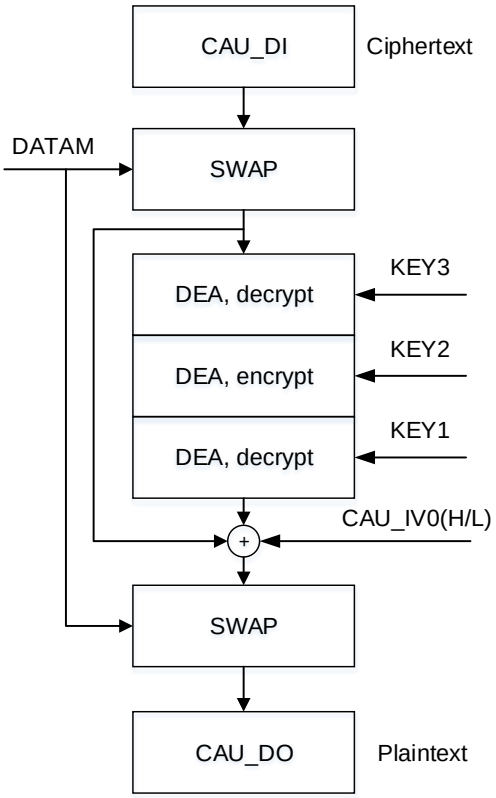
**Figure 13-6. DES / TDES CBC encryption**




### DES / TDES CBC decryption

In DES / TDES CBC decryption, when the TDES algorithm is configured, the first ciphertext block is used directly after data swapping according to the data type, it is read in the DEA and decrypted using KEY3. The output is fed back directly to next DEA and then encrypted using KEY2. After that, the output is fed back directly to the last DEA and decrypted with KEY1. The first result of above process is then XORed with the initialization vector which is the same as that used during encryption. At the same time, the first ciphertext is then used as the next initialization vector and exclusive-ORed with the next result after DEA blocks. The above operations are repeated until the last ciphertext block is decrypted. Note if the ciphertext message does not consist of an integral number of data blocks, the final partial data block should be decrypted in a specified manner same to that in encryption. At last, the output plaintext is also obtained after data swapping according to the data type. When the DES algorithm is configured, the state and process of the second and third block of DEA should also be omitted. The procedure of DES / TDES CBC mode decryption is illustrated in ***Figure 13-7. DES / TDES CBC decryption***.

**Figure 13-7. DES / TDES CBC decryption**




## 13.4.2. AES cryptographic acceleration processor

The AES cryptographic acceleration processor consists of three components, including the AES algorithm (AEA), multiple keys and the initialization vectors or Nonce.

Three lengths of AES keys are supported: 128, 192 and 256 bits, and different initialization vectors or nonce are used depends on the operation mode.

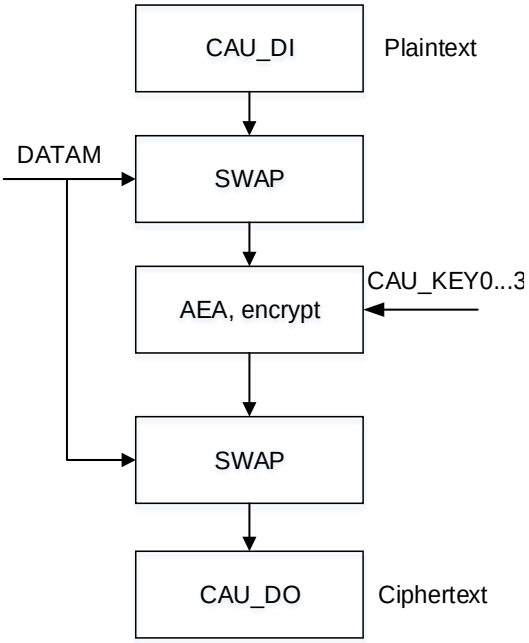
The AES key is used as [KEY3 KEY2] when the key size is configured as 128, [KEY3 KEY2 KEY1] when the key size is configured as 192 and [KEY3 KEY2 KEY1 KEY0] when the key size is configured as 256.

The thorough explanation of the key used in the AES is provided in FIPS PUB 197 (November 26, 2001), and the explanation process is omitted in this manual.

### AES-ECB mode encryption

The 128-bit input plaintext is first obtained after data swapping according to the data type. The input data block is read in the AEA and encrypted using the 128, 192 or 256 -bit key. The output after above process is then swapped back according to the data type again, and a 128-bit ciphertext is produced and stored in the out FIFO. The procedure of AES ECB mode encryption is illustrated in ***Figure 13-8. AES ECB encryption***.

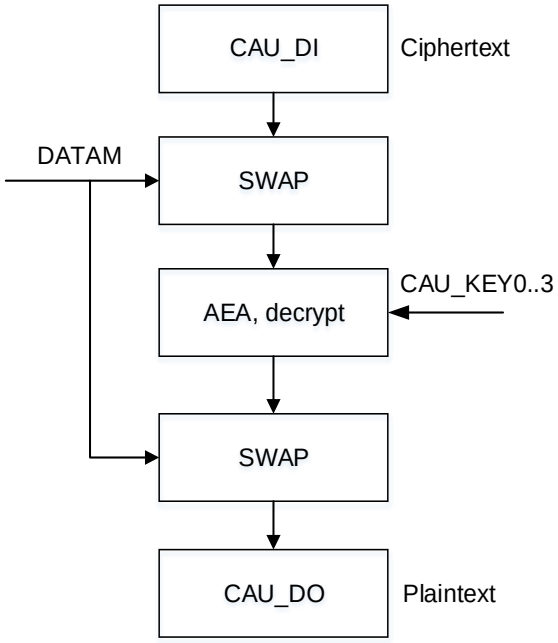
**Figure 13-8. AES ECB encryption**




### AES-ECB mode decryption

First of all, the key derivation must be completed to prepare the decryption keys, the input key of the key schedule is the same to that used in encryption. The last round key obtained from the above operation is then used as the first round key in the decryption. After the key derivation, the 128-bit input ciphertext is first obtained after data swapping according to the data type. The input data block is read in the AEA and decrypted using keys prepared above. The output is then swapped back according to the data type again, and a 128-bit plaintext is produced. The procedure of AES ECB mode decryption is illustrated in ***Figure 13-9. AES ECB decryption***.

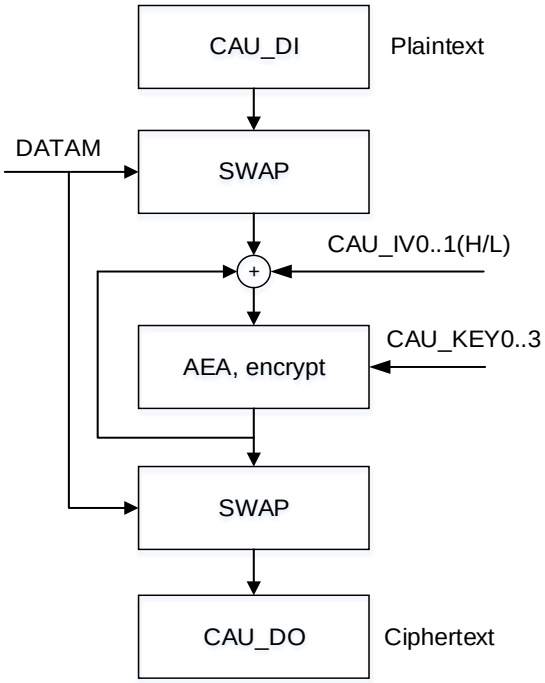
**Figure 13-9. AES ECB decryption**




### AES-CBC mode encryption

The input data of the AEA block in CBC mode consists of two aspects: the input plaintext after data swapping according to the data type, and the initialization vectors. The XOR result of the swapped plaintext data block and the 128-bit initialization vector CAU_IV0..1 is read in the AEA and encrypted using the 128-, 192-, 256-bit key. The result is then used as the next initialization vector and exclusive-ORed with the next plaintext data block to process next encryption. The above operations are repeated until the last plaintext block is encrypted. Note that the final partial data block should be encrypted in a specified manner if the plaintext message does not consist of an integral number of data blocks. At last, the output ciphertext is also obtained after data swapping according to the data type. The procedure of AES CBC mode encryption is illustrated in ***Figure 13-10. AES CBC encryption***.

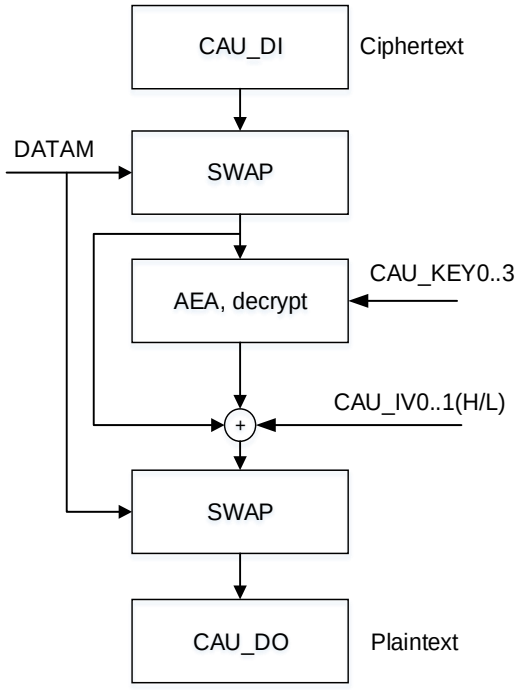
**Figure 13-10. AES CBC encryption**




### AES-CBC mode decryption

Similar to that in AES-ECB mode decryption, the key derivation also must be completed first to prepare the decryption keys, the input of the key schedule should be the same to that used in encryption. The last round key obtained from the above operation is then used as the first round key in the decryption. After the key derivation, the 128-bit input ciphertext is first obtained after data swapping according to the data type. The input data block is read in the AEA and decrypted using keys prepared above. At the same time, the first ciphertext is then used as the next initialization vector and exclusive-ORed with the next result after AEA blocks (The first initialization is obtained directly from the CAU_IV0..1 registers). The above operations are repeated until the last ciphertext block is decrypted. Note if the ciphertext message does not consist of an integral number of data blocks, the final partial data block should be decrypted in a specified manner same to that in encryption. At last, the output plaintext is also obtained after data swapping according to the data type. The procedure of AES CBC mode decryption is illustrated in ***Figure 13-11. AES CBC decryption***.

**Figure 13-11. AES CBC decryption**




### AES-CTR mode

In counter mode, a counter is used in addition with a nonce value to be encrypted and decrypted in AEA, and the result will be used for the XOR operation with the plaintext or the ciphertext. As the counter is incremented from the same initialized value for each block in encryption and decryption, the key schedule during the encryption and decryption are the same. Then decryption operation acts exactly in the same way as the encryption operation. Only the 32-bit LSB of the 128-bit initialization vector represents the counter, which means the other 96 bits are unchanged during the operation, and the initial value should be set to 1. Nonce is 32-bit single-use random value and should be updated to each communication block. And the 64-bit initialization vector is used to ensure that a given value is used only once for a given key. ***Figure 13-12. Counter block structure*** illustrates the counter block structure and ***Figure 13-13. AES CTR encryption/decryption*** shows the AES CTR encryption / decryption.

**Figure 13-12. Counter block structure**

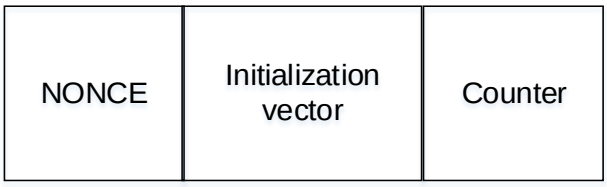
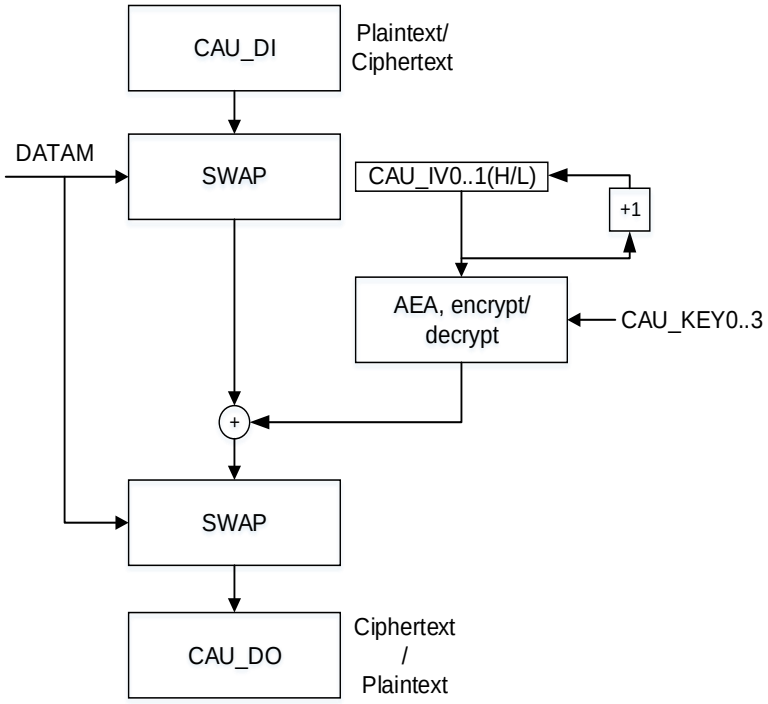


**Figure 13-13. AES CTR encryption/decryption**




## AES-GCM mode

The AES Galois / counter mode (GCM) can be used to encrypt or authenticate message, and then ciphertext and tag can be obtained. This algorithm is based on AES CTR mode to ensure confidentiality. A multiplier over a fixed finite field is used to generate the tag.

In this mode, four steps are required to perform an encryption/decryption:

1. GCM prepare phase

   The hash key is calculated and saved internally to be used later.

   (a) Clear the CAUEN bit to make sure CAU is disabled.
   (b) Configure the ALGM[3:0] bits to '1000'.
   (c) Configure GCM_CCMPH[1:0] bits to '00'.
   (d) Configure key registers and initialization vectors.
   (e) Enable CAU by writing 1 to CAUEN bit.
   (f) Wait until CAUEN bit is cleared by hardware, and then enable CAU again for following phases.

2. GCM AAD (additional authenticated data) phase

   This phase must be performed after GCM prepare phase and also precede the encryption / decryption phase. In this phase, data is authenticated but not protected.

   (g) Configure GCM_CCMPH[1:0] bits to '01'.
   (h) Write data into CAU_DI register, INF and IEM flags can be used to determine if the input FIFO can receive data. The size of the AAD must be a multiple of 128bits. DMA

can also be used.

(i) Repeat (h) until all AAD data are supplied, wait until BUSY bit is cleared.

3. GCM encryption / decryption phase

This phase must be performed after GCM AAD phase. In this phase, the message is authenticated and encrypted / decrypted.

(j) Configure GCM_CCMPH[1:0] bits to '10'.
(k) Configure the computation direction in CAUDIR.
(l) Write data into CAU_DI register, INF and IEM flags can be used to determine if the input FIFO can receive data. ONE and OFU flags can be used to check if the output FIFO is not empty. If so, read the CAU_DO register. DMA can also be used.
(m) Repeat (l) step until all payload blocks are processed.

4. GCM tag phase

In this phase, the final authentication tag is generated.

(n) Configure GCM_CCMPH[1:0] bits to '11'.
(o) Write the input into the CAU_DI register, 4 times write operation is needed. The input consists of the AAD data size (64bits) and the payload data size (64bits).
(p) Wait until the ONE flag is set to 1, and then read CAU_DO 4 times. The output corresponds to the authentication tag.
(q) Disable the CAU.

**Note:** The key should be prepared at the beginning when a decryption is performed.

### AES-GMAC mode

The AES Galois message authentication code mode is also supported to authenticate the message. It is processing based on the AES-GCM mode, while the encryption / decryption phase is by-passed.

### AES-CCM mode

The AES combined cipher machine mode, which is similar to AES-GCM mode, also allows encrypting and authenticating message. It is also based on AES-CTR mode to ensure confidentiality. In this mode, AES-CBC is used to generate a 128-bit tag.

The CCM standard (RFC 3610 Counter with CBC-MAC (CCM) dated September 2003) defines particular encoding rules for the first authentication block (B0 in the standard). In particular, the first block includes flags, a nonce and the payload length expressed in bytes. The CCM standard specifies another format, called A or counter, for encryption / decryption. The counter is incremented during the encryption / decryption phase and its 32 LSB bits are initialized to '1' during the tag generation (A0 packet in the CCM standard).

**Note:** The formatting operation of B0 packet should be handled by software.

In this mode, four steps are required to perform an encryption / decryption:

1. CCM prepare phase

   In this phase, B0 packet (the first packet) is programmed into the CAU_DI register. CAU_DO never contain data in this phase.

   (a) Clear the CAUEN bit to make sure CAU is disabled.
   (b) Configure the ALGM[3:0] bits to '1001'.
   (c) Configure GCM_CCMPH[1:0] bits to '00'.
   (d) Configure key registers and initialization vectors.
   (e) Enable CAU by writing 1 to CAUEN bit.
   (f) Program the B0 packet into the CAU_DI.
   (g) Wait until CAUEN is cleared by hardware, and then enable CAU again for following phases.

2. CCM AAD (additional authenticated data) phase

   This phase must be performed after CCM prepare phase and also precede the encryption / decryption phase. In this phase, CAU_DO never contain data.

   This phase can be by-passed if there is no additional authenticated data.

   (h) Configure GCM_CCMPH[1:0] bits to '01'
   (i) Write data into CAU_DI register, INF and IEM flags can be used to determine if the input FIFO can receive data. The size of the AAD must be a multiple of 128 bits. DMA can also be used.
   (j) Repeat (i) until all additional authenticated data are supplied, wait until BUSY bit is cleared

3. CCM encryption / decryption phase

   This phase must be performed after CCM AAD phase. In this phase, the message is authenticated and encrypted / decrypted.

   Like GCM, the CCM chaining mode can be applied on a message composed only by plaintext authenticated data (that is, only AAD, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM / GMAC).

   (k) Configure GCM_CCMPH[1:0] bits to '10'
   (l) Configure the computation direction in CAUDIR
   (m) Write data into CAU_DI register, INF and IEM flags can be used to determine if the input FIFO can receive data. ONE and OFU flags can be used to check if the output FIFO is not empty. If so, read the CAU_DO register. DMA can also be used.
   (n) Repeat (m) step until all payload blocks are processed.

4. CCM tag phase

   In this phase, the final authentication tag is generated.

   (o) Configure GCM_CCMPH[1:0] bits to '11'
   (p) Write the 128 bit input into the CAU_DI register, 4 times of write operation to CAU_DI is needed. The input is the A0 value.

(q) Wait until the ONE flag is set to 1, and then read CAU_DO 4 times. The output corresponds to the authentication tag.

(r) Disable the CAU

#### AES-CFB mode

The Cipher Feedback (CFB) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and the decryption process is similar to the encryption described before.

#### AES-OFB mode

The Output Feedback (OFB) mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and the decryption process is similar to the encryption described before.

## 13.5. Operating modes

#### Encryption

1. If users want to use AES key from EFUSE, go to step 2, otherwise, go to step 5.
2. Write AES keys (in little endian) to EFUSE_AES_KEYx registers in order, check whether the CRC values are right and then write the AES keys to EFUSE.
3. Set AESEN bit in EFUSE_USER_CTL register and then write this bit to efuse.
4. Make sure AESNCAU bit in the EFUSE_MCU_RSV register is 0.
5. Disable the CAU by resetting the CAUEN bit in the CAU_CTL register.
6. If the AES key comes from registers of the CAU, skip this step. If the AES key comes from efuse, make sure that the key has been written to efuse and set KEY_SEL bit.
7. Select and configure the key length with the KEYM bits in the CAU_CTL register if AES algorithm is chosen. If the AES key comes from efuse, skip this step.
8. Reset KEY_SEL bit in the CAU_CTL register then configure the CAU_KEY0..3(H / L) registers according to the algorithm. If the AES key comes from efuse, skip this step.
9. Configure the DATAM bit in the CAU_CTL register to select the data swapping type.
10. Configure the algorithm (DES / TDES / AES) and the chaining mode (ECB / CBC / CTR / GCM / GMAC / CCM / CFB / OFB) by writing the ALGM[3:0] bit in the CAU_CTL register.
11. Configure the encryption direction by writing 0 to the CAUDIR bit in the CAU_CTL register.
12. Configure the initialization vectors by writing the CAU_IV0..1 registers.
13. Flush the input FIFO and output FIFO by configure the FFLUSH bit in the CAU_CTL register when CAUEN is 0.
14. Enable the CAU by set the CAUEN bit as 1 in the CAU_CTL register.

15. If the INF bit in the CAU_STAT0 register is 1, then write data blocks into the CAU_DI register. The data can be transferred by DMA / CPU during interrupts / no DMA or interrupts.
16. Wait for ONE bit in the CAU_STAT0 register is 1 then read the CAU_DO register. The output data can also be transferred by DMA / CPU during interrupts / no DMA or interrupts.
17. Repeat steps 15, 16 until all data blocks has been encrypted.

**Note:** When users use AES key from EFUSE (KEY_SEL is set), AES algorithm mode is forced AES-CTR mode (ALGM[3:0] is forced to 0b0110) and AES key length is forced to 128-bits.

### Decryption

1. Disable the CAU by resetting the CAUEN bit in the CAU_CTL register.
2. Select and configure the key length with the KEYM bits in the CAU_CTL register if AES algorithm is chosen.
3. Reset KEY_SEL bit in the CAU_CTL register then configure the CAU_KEY0..3(H / L) registers according to the algorithm. If the key comes from efuse, make sure that the key has been written to EFUSE and AESNCAU bit in the EFUSE_MCU_RSV register is 0, then set KEY_SEL bit.
4. Configure the DATAM bit in the CAU_CTL register to select the data swapping type.
5. Configure the ALGM[3:0] bits to “0111” in the CAU_CTL register to complete the key derivation.
6. Enable the CAU by set the CAUEN bit as 1.
7. Wait until the BUSY and CAUEN bit return to 0 to make sure that the decryption keys are prepared.
8. Configure the algorithm (DES / TDES / AES) and the chaining mode (ECB / CBC / CTR / GCM / GMAC / CCM / CFB / OFB) by writing the ALGM[3:0] bit in the CAU_CTL register.
9. Configure the decryption direction by writing 1 to the CAUDIR bit in the CAU_CTL register.
10. Configure the initialization vectors by writing the CAU_IV0..1 registers.
11. Flush the input FIFO and output FIFO by configure the FFLUSH bit in the CAU_CTL register when CAUEN is 0.
12. Enable the CAU by set the CAUEN bit as 1 in the CAU_CTL register.
13. If the INF bit in the CAU_STAT0 register is 1, then write data blocks into the CAU_DI register. The data can be transferred by DMA / CPU during interrupts/no DMA or interrupts.
14. Wait for ONE bit in the CAU_STAT0 register is 1, then read the CAU_DO register. The output data can also be transferred by DMA / CPU during interrupts / no DMA or interrupts.
15. Repeat steps 13, 14 until all data blocks has been decrypted.

### Data append

For GCM payload encryption or CCM payload decryption, CAU supports non 128 bit integer

multiple data block processing. When the last data block is less than 128bit, use '0' to fill the remaining bits, and then configure the number of bytes to be filled in the NBPILB bitfield of the CAU_CTL register. AES will automatically remove the the number of filled pads and encrypt / decrypt it. It should be noted that the NBPILB[3:0] bitfield should be configured after the encryption / decryption of the penultimate data block is completed.

## 13.6. CAU DMA interface

The DMA can be used to transfer data blocks with the interface of the cryptographic acceleration unit. The operations can be controlled by the CAU_DMAEN register. DMAIEN is used to enable the DMA request during the input phase, then a word is written into CAU_DI from DMA. DMAOEN is used to enable the DMA request during the output phase, then a word is read from the CAU.

Single and Burst transfers are both supported to ensure the data transfer if the number of words is not an integral multiple of burst size. Note the DMA controller should be configured to perform burst of 4 words or less to make sure no data will be lost. DMA channel for output data has a higher priority than that channel for input data so that the output FIFO can be empty earlier than that the input FIFO is full.

## 13.7. CAU interrupts

There are two types of interrupt registers in CAU, which are CAU_STAT1 and CAU_INTF. In CAU, the interrupt is used to indicate the situation of the input and output FIFO.

Any of input and output FIFO interrupt can be enabled or disabled by configuring the Interrupt Enable register CAU_INTEN. Value 1 of the register enable the interrupts.

### Input FIFO interrupt

The input FIFO interrupt is asserted when the number of words in the input FIFO is less than four words, then ISTA is asserted. And if the input FIFO interrupt is enabled by IINTEN with a 0 value, the IINTF is also asserted. Note if the CAUEN is low, then the ISTA and IINTF are also always low.

### Output FIFO interrupt

The output FIFO interrupt is asserted when the number of words in the output FIFO is more than one words, then OSTA is asserted. And if the output FIFO interrupt is enabled by OINTEN with a 0 value, the OINTF is also asserted. Note Unlike that of Input FIFO interrupt, the value of CAUEN will never affect the situation of OSTA and OINTF.

## 13.8. CAU suspended mode

It is possible to suspend a data block if another new data block with a higher priority needs to be processed in CAU. The following steps can be performed to complete the encryption / decryption acceleration of the suspended data blocks.

**When DMA transfer is used:**

1. Stop the current input transfer. Clear the DMAIEN bit in the CAU_DMAEN register.
2. When it is DES or AES, wait until both the input and output FIFO are both empty if the input FIFO is not empty (IEM = 0), then write a word of data into CAU_DI register, do as so until the IEM is checked to be 1, then wait until the BUSY bit is cleared, so that the next data block will not be affected by the last one. Case of TDES is similar to that of AES except that it does not need to wait until the input FIFO is empty.
3. Stop the output transfer by clearing the DMAOEN bit in the CAU_DMAEN register. And disable the CAU by clearing the CAUEN bit in the CAU_CTL register.
4. Save the configuration, including the key size, data type, operation mode, direction, GCM CCM phase and the key values. When it is CBC, CTR, GCM, GMAC, CCM, CFB or OFB chaining mode, the initialization vectors should also be stored. When it is GCM, GMAC or CCM mode, the context switch CAU_GCMCCMCTXSx (x=0..7) and CAU_GCMCTXSx (x=0..7) registers should also be stored.
5. Configure and process the new data block.
6. Restore the process before. Configure the CAU with the parameters stored before, and prepare the key and initialization vectors, and the context switch registers CAU_GCMCCMCTXSx (x=0..7) and CAU_GCMCTXSx (x=0..7) should also be restored. Then enable CAU by setting the CAUEN bit in the CAU_CTL register.

**When data transfer is done by CPU access to CAU_DI and CAU_DO:**

1. When the data transfer is done by CPU access, then wait for the fourth read of the CAU_DO register and before the next CAU_DI write access so that the message is suspended at the end of a block processing.
2. Disable the CAU by clearing the CAUEN bit in the CAU_CTL register.
3. Save the configuration, including the key size, data type, operation mode, direction, GCM CCM phase and the key values. When it is CBC, CTR, GCM, GMAC, CCM, CFB or OFB chaining mode, the initialization vectors should also be stored. When it is GCM, GMAC or CCM mode, the context switch CAU_GCMCCMCTXSx (x=0..7) and CAU_GCMCTXSx (x=0..7) registers should also be stored.
4. Configure and process the new data block.
5. Restore the process before. Configure the CAU with the parameters stored before, and prepare the key and initialization vectors, and the context switch registers CAU_GCMCCMCTXSx (x=0..7) and CAU_GCMCTXSx (x=0..7) should also be restored. Then enable CAU by setting the CAUEN bit in the CAU_CTL register.

## 13.9. Register definition

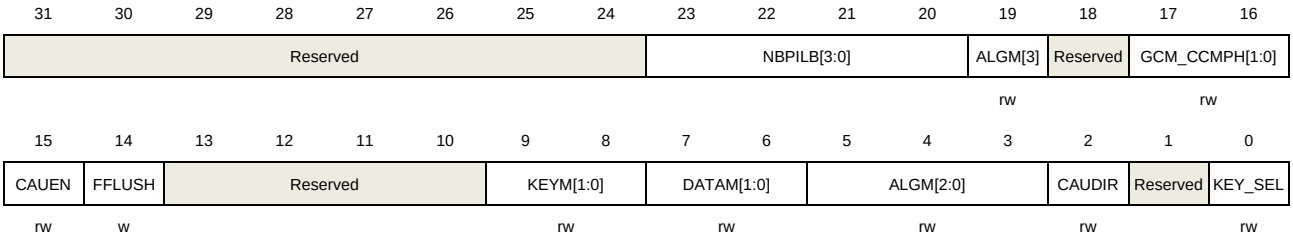
CAU base address: 0x4802 1000

### 13.9.1. Control register (CAU_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:20 | NBPILB[3:0] | Number of bytes padding in last block of payload<br>0000: all bytes are valid (no padding)<br>0001: one padding byte of last block<br>...<br>1111: 15 padding bytes of last block |
| 19 | ALGM[3] | Encryption / decryption algorithm mode bit 3 |
| 18 | Reserved | Must be kept at reset value. |
| 17:16 | GCM_CCMPH[1:0] | GCM CCM phase<br>00: prepare phase<br>01: AAD phase<br>10: encryption / decryption phase<br>11: tag phase |
| 15 | CAUEN | CAU Enable<br>0: CAU is disabled<br>1: CAU is enabled<br>**Note:** the CAUEN can be cleared automatically when the key derivation (ALGM=0111b) is finished or the AES-GCM or AES-CCM initial phase finished. |
| 14 | FFLUSH | Flush FIFO<br>0: No effect<br>1: When CAUEN=1, flush the input and output FIFO<br>Reading this bit always returns 0 |

| | | |
|---|---|---|
| 13:10 | Reserved | Must be kept at reset value. |
| 9:8 | KEYM[1:0] | AES key size mode configuration, must be configured when BUSY=0<br>00: 128-bit key length<br>01: 192-bit key length<br>10: 256-bit key length<br>11: never use |
| 7:6 | DATAM[1:0] | Data swapping type mode configuration, must be configured when BUSY=0<br>00: No swapping<br>01: Half-word swapping<br>10: Byte swapping<br>11: Bit swapping |
| 5:3 | ALGM[2:0] | Encryption / decryption algorithm mode bit 0 to bit 2<br>These bits and bit 19 of CAU_CTL must be configured when BUSY=0<br>0000: TDES-ECB with CAU_KEY1, 2, 3.<br>Initialization vectors (CAU_IV0..1) are not used<br>0001: TDES-CBC with CAU_KEY1, 2, 3.<br>Initialization vectors (CAU_IV0) is used to XOR with data blocks<br>0010: DES-ECB with only CAU_KEY1<br>Initialization vectors (CAU_IV0..1) are not used<br>0011: DES-CBC with only CAU_KEY1<br>Initialization vectors (CAU_IV0) is used to XOR with data blocks<br>0100: AES-ECB with CAU_KEY0, 1, 2, 3.<br>Initialization vectors (CAU_IV0..1) are not used<br>0101: AES-CBC with CAU_KEY0, 1, 2, 3.<br>Initialization vectors (CAU_IV0..1) are used to XOR with data blocks<br>0110: AES_CTR with CAU_KEY0, 1, 2, 3.<br>Initialization vectors (CAU_IV0..1) are used to XOR with data blocks<br>In this mode, encryption and decryption are same, then the CAUDIR is disregarded.<br>0111: AES key derivation for decryption mode. The input key must be same to that used in encryption. The BUSY bit is set until the process has been finished, and CAUEN is then cleared.<br>1000: Galois Counter Mode (GCM). This algorithm mode is also used for GMAC algorithm.<br>1001: Counter with CBC-MAC (CCM).<br>1010: Cipher Feedback (CFB) mode<br>1011: Output Feedback (OFB) mode |
| 2 | CAUDIR | CAU direction, must be configured when BUSY=0<br>0: encryption<br>1: decryption |
| 1 | Reserved | Must be kept at reset value. |
| 0 | KEY_SEL | Key select |

0: Use key from CAU_KEY0..3(H/L)

1:Use key from efuse

### 13.9.2. Status register 0 (CAU_STAT0)

Address offset: 0x04

Reset value: 0x0000 0003

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | BUSY | OFU | ONE | INF | IEM |
| | | | | | | | | | | | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | BUSY | Busy bit<br>0: No processing. This is because:<br>- CAU is disabled by CAUEN=0 or the processing has been completed.<br>- No enough data or no enough space in the input / output FIFO to perform a data block<br>1: CAU is processing data or key derivation. |
| 3 | OFU | Output FIFO is full<br>0: Output FIFO is not full<br>1: Output FIFO is full |
| 2 | ONE | Output FIFO is not empty<br>0: Output FIFO is empty<br>1: Output FIFO is not empty |
| 1 | INF | Input FIFO is not full<br>0: Input FIFO is full<br>1: Input FIFO is not full |
| 0 | IEM | Input FIFO is empty<br>0: Input FIFO is not empty<br>1: Input FIFO is empty |

### 13.9.3. Data input register (CAU_DI)

Address offset: 0x08

Reset value: 0x0000 0000

The data input register is used to transfer plaintext or ciphertext blocks into the input FIFO for processing. The MSB is firstly written into the FIFO and the LSB is the last one. If the CAUEN is 0 and the input FIFO is not empty, when it is read, then the first data in the FIFO is popped out and returned. If the CAUEN is 1, the returned value is undefined. Once it is read, then the FIFO must be flushed.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DI[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DI[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DI[31:0] | Data input<br>Write these bits will write data to input FIFO, read these bits will return input FIFO value if CAUEN is 0, or it will return an undefined value |

### 13.9.4. Data output register (CAU_DO)

Address offset: 0x0C
Reset value: 0x0000 0000

The data output register is a read only register. It is used to receive plaintext or ciphertext results from the output FIFO. Similar to CAU_DI, the MSB is read at first while the LSB is read at last.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DO[31:0] | Data output<br>These bits are read only, read these bits return output FIFO value. |

### 13.9.5. DMA enable register (CAU_DMAEN)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | DMAOEN | DMAIEN |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | DMAOEN | DMA output enable<br>0: DMA for OUT FIFO data is disabled<br>1: DMA for OUT FIFO data is enabled |
| 0 | DMAIEN | DMA input enable<br>0: DMA for IN FIFO data is disabled<br>1: DMA for IN FIFO data is enabled |

### 13.9.6. Interrupt enable register (CAU_INTEN)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | OINTEN | IINTEN |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | OINTEN | OUT FIFO interrupt enable<br>0: OUT FIFO interrupt is disable<br>1: OUT FIFO interrupt is enable |
| 0 | IINTEN | IN FIFO interrupt enable<br>0: IN FIFO interrupt is disable<br>1: IN FIFO interrupt is enable |

### 13.9.7. Status register 1 (CAU_STAT1)

Address offset: 0x18

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | OSTA | ISTA |
| | | | | | | | | | | | | | | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | OSTA | OUT FIFO interrupt status<br>0: OUT FIFO interrupt status not pending<br>1: OUT FIFO interrupt status pending |
| 0 | ISTA | IN FIFO interrupt status<br>0: IN FIFO interrupt not pending<br>1: IN FIFO interrupt flag pending |

## 13.9.8. Interrupt flag register (CAU_INTF)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | OINTF | IINTF |
| | | | | | | | | | | | | | | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | OINTF | OUT FIFO enabled interrupt flag<br>0: OUT FIFO Interrupt not pending<br>1: OUT FIFO Interrupt pending |
| 0 | IINTF | IN FIFO enabled interrupt flag<br>0: IN FIFO Interrupt not pending<br>1: IN FIFO Interrupt pending when CAUEN is 1 |

### 13.9.9. Key registers (CAU_KEY0..3(H / L))

Address offset: 0x20 to 0x3C
Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit), and all of them must be written when BUSY is 0.

In DES mode, only CAU_KEY1 is used.

In TDES mode, CAU_KEY1, CAU_KEY2 and CAU_KEY3 are used.

In AES-128 mode, KEY2H[31:0] || KEY2L[31:0] is used as AES_KEY[0:63], and KEY3H[31:0] || KEY3L[31:0] is used as AES_KEY[64:127].

In AES-192 mode, KEY1H[31:0] || KEY1L[31:0] is used as AES_KEY[0:63], KEY2H[31:0] || KEY2L[31:0] is used as AES_KEY[64:127], and KEY3H[31:0] || KEY3L[31:0] is used as AES_KEY[128:191].

In AES-256 mode, KEY0H[31:0] || KEY0L[31:0] is used as AES_KEY[0:63], KEY1H[31:0] || KEY1L[31:0] is used as AES_KEY[64:127], KEY2H[31:0] || KEY2L[31:0] is used as AES_KEY[128:191], and KEY3H[31:0] || KEY3L[31:0] is used as AES_KEY[192:255].

**Note**: "||" is a concatenation operator. For example, X || Y denotes the concatenation of two bit strings X and Y.

#### CAU_KEY0H

Address offset: 0x20
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY0H[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY0H[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

#### CAU_KEY0L

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY0L[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY0L[15:0] | | | | | | | | | | | | | | | |

w

### CAU_KEY1H

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY1H[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEY1H[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

### CAU_KEY1L

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY1L[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEY1L[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

### CAU_KEY2H

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY2H[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| KEY2H[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

### CAU_KEY2L

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY2L[31:16] | | | | | | | | | | | | | | | |

w

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY2L[15:0] | | | | | | | | | | | | | | | |

w

### CAU_KEY3H

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY3H[31:16] | | | | | | | | | | | | | | | |

w

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY3H[15:0] | | | | | | | | | | | | | | | |

w

### CAU_KEY3L

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY3L[31:16] | | | | | | | | | | | | | | | |

w

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY3L[15:0] | | | | | | | | | | | | | | | |

w

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | KEY0...3(H / L) | The key for DES, TDES, AES |

## 13.9.10. Initial vector registers (CAU_IV0..1(H / L))

Address offset: 0x40 to 0x4C

Reset value: 0x0000 0000

This registers have to be accessed by word (32-bit), and all of them must be written when BUSY is 0.

In DES / TDES mode, IV0H is the leftmost bits, and IV0L is the rightmost bits of the initialization vectors.

In AES mode, IV0H is the leftmost bits, and IV1L is the rightmost bits of the initialization vectors.

### CAU_IV0H

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV0H[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV0H[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

### CAU_IV0L

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV0L[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV0L[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

### CAU_IV1H

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV1H[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV1H[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

### CAU_IV1L

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV1L[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IV1L[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | IV0...1(H / L) | The initialization vector for DES, TDES, AES |

### 13.9.11. GCM or CCM mode context switch register x (CAU_GCMCCMCTXSx) (x=0..7)

Address offset: 0x50 to 0x6C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTXx[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTXx[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CTXx[31:0] | The internal status of the CAU core. Read and save the register data when a high-priority task is coming to be processed, and restore the saved data back to the registers to resume the suspended processing.<br>**Note:** These registers are used only when GCM, GMAC or CCM mode is selected. |

### 13.9.12. GCM mode context switch register x (CAU_GCMCTXSx) (x=0..7)

Address offset: 0x70 to 0x8C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTXx[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTXx[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CTXx[31:0] | The internal status of the CAU core. Read and save the register data when a high-priority task is coming to be processed, and restore the saved data back to the registers to resume the suspended processing. |

**Note:** These registers are used only when GCM or GMAC mode is selected.

# 14. Hash Acceleration Unit (HAU) (For GD32H737_757_759)

## 14.1. Overview

The hash acceleration unit is used for information security. The secure hash algorithm (SHA-1, SHA-224, SHA-256), the message-digest algorithm (MD5) and the keyed-hash message authentication code (HMAC) algorithm are supported for various applications. The digest will be computed and the length is 160 / 224 / 256 / 128 bits for a message up to ($2^{64}$ - 1) bits computed by SHA-1, SHA-224, SHA-256 and MD5 algorithms respectively. In HMAC algorithm, SHA-1, SHA-224, SHA-256 or MD5 will be called twice as hash functions and authenticating messages can be produced.

The HAU is fully compliant implementation of the following standards:

- Federal Information Processing Standards Publication 180-4(FIPS PUB 180-4).
- Secure Hash Standard specifications (SHA-1, SHA-224, SHA-256).
- Internet Engineering Task Force Request for Comments number 1321 (IETF RFC 1321) specifications (MD5).

## 14.2. Characteristics

- 32-bit AHB slave peripheral.
- High performance of computation of hash algorithms.
- Little-endian data representation.
- Multiple data types are supported, including no swapping, half-word swapping, byte swapping, and bit swapping with 32-bit data words.
- Automatic data padding to fill the 512-bit message block for digest computation.
- DMA transfer is supported.
- Hash / HMAC process suspended mode.

## 14.3. HAU data type

The hash acceleration unit receives data words of 32 bits at a time, while they are processed in 512-bits blocks. For each input word, according to the data type, the data could be bit / byte / half-word / no swapped before they are transferred into the hash acceleration core. The same swapping operation should be also performed on the core output data before they are collected. Note the least-significant data always occupies the lowest address location no matter which data type is configured, because the system memory is little-endian. However, the computation of SHA-1, SHA-224 and SHA-256 are big-endian.

*Figure 14-1. DATAM No swapping and Half-word swapping* and *Figure 14-2. DATAM*

*Byte swapping and Bit swapping* illustrate the data swapping according to different data types.

**Figure 14-1. DATAM No swapping and Half-word swapping**

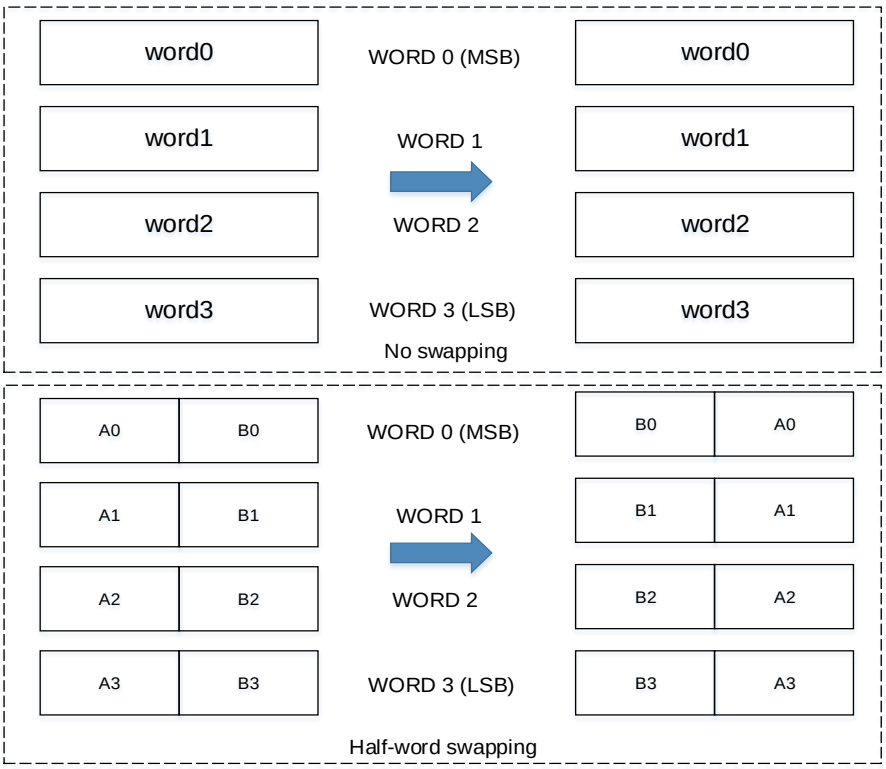
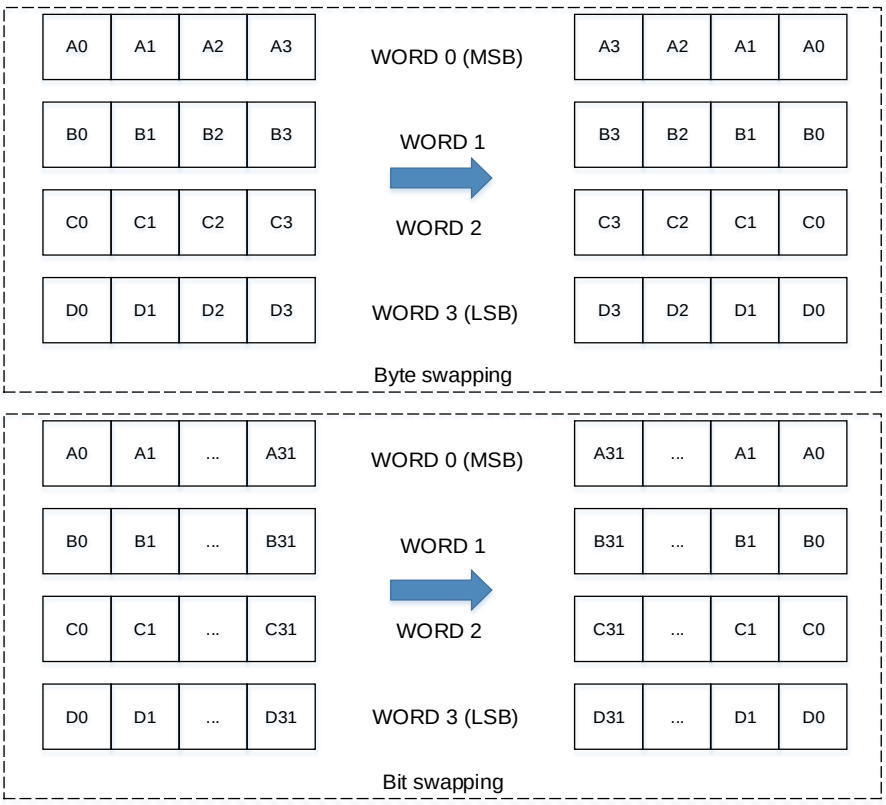



**Figure 14-2. DATAM Byte swapping and Bit swapping**

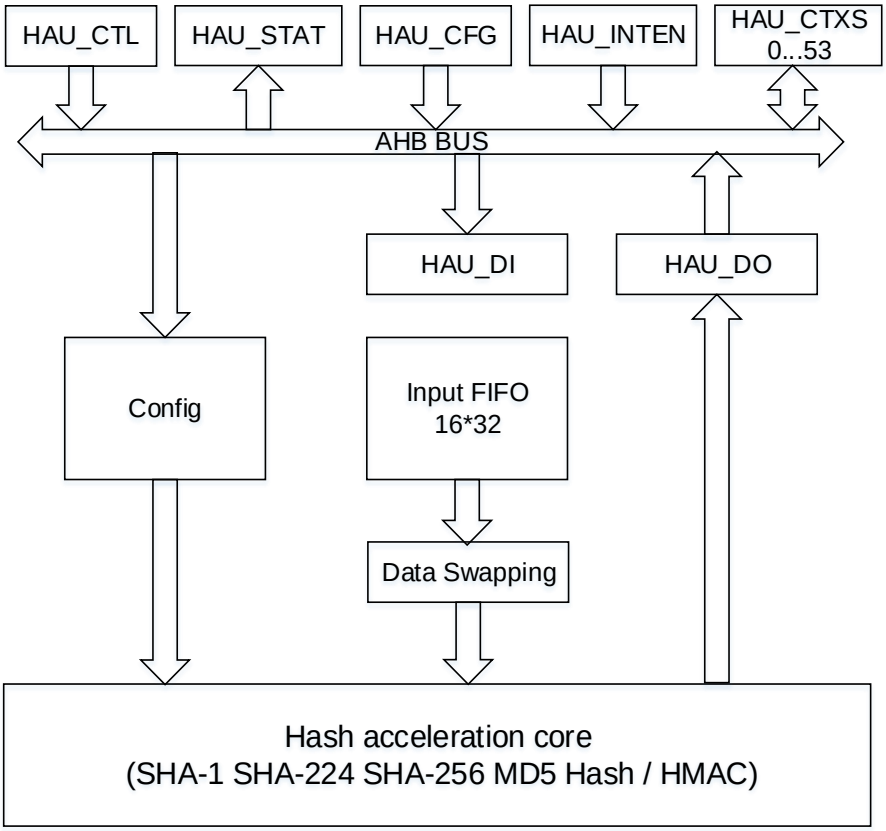


## 14.4. HAU core

The hash acceleration unit is used to compute condensed information of input messages with secure hash algorithms. The digest result has a length of 160 / 224 / 256 / 128 bits for a message up to ($2^{64}$-1) bits computed by SHA-1, SHA-224, SHA256 and MD5 algorithms respectively. It can be used to generate or verify the signature of a message with a higher efficiency because of the much simpler of the information.

A message which need to be processed in the HAU should be considered as bit information. And the length is the number of bits of the message. The information security is ensured because that, to find the original message using the digest is computationally impossible and, the result will be completely different with any change to the input message.

**Figure 14-3. HAU block diagram**




### 14.4.1. Automatic data padding

The input message should be padded first so that the number of bits in the input of the HAU core can be an integral multiple of 512. First of all, a “1” is added to follow the last bit of the input message, and then several “0” should be padded to ensure the result modulo 512 is 448, at last, a 64-bit length information of input is added.

After the message padding is correctly performed, the VBL bits in the HAU_CFG register is configured as the 64-bit length value above, and CALEN bit in the HAU_CFG register can be

set 1 to start the calculation of the digest of the last block.

Data Padding Example: The input message is “HAU”, which ASCII hexadecimal code is:

484155

Then the VBL bits in the HAU_CFG register is set as decimal 24 because of the valid bit length. A “1” is added at bit location 24 then, and several “0” are padded so that the result modulo 512 is 448, the hexadecimal result is as follows:

48415580 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000

After that, a 64-bit length information of the input message is padded, which hexadecimal value is 18, and the final result will be:

48415580 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000018

### 14.4.2. Digest computing

After data padding, for each block calculation of HAU, 512 bits are written into the HAU core by DMA or CPU. To start the processing of the HAU core, the peripheral must obtain the information as to whether the HAU_DI register contains the last bits of the message or not. This can be confirmed with the status of the input FIFO and the HAU_DI register.

When DMA is used to transfer data:

The status of the block transfer is automatically interpreted with the information from the DMA controller. And padding and digest computation are performed automatically as if CALEN bit in the HAU_CFG register is set as 1.

**Note:** If hash message is large files and multiple DMA transfers are needed, then MDS bit should be set as 1. And the VBL bits need to be set before the transfer. The CALEN bit is not set automatically after an intermediate DMA transfer completed. Only when the last DMA transfer is processing, the MDS bit is cleared so that the CALEN bit is automatically set after data transferring.

Otherwise, the MDS bit is set as 0. And the CALEN bit is set automatically after a DMA transfer. Also, VBL bits need to set before the DMA transfer.

When CPU is used to transfer data without DMA:

- The intermediate block computing can be started when HAU_DI is filled with another new word of the next block.
- The last block computing can be started when CALEN bit in the HAU_CFG register is 1.

### 14.4.3. Hash mode

The hash mode is selected when the HMS bit in the HAU_CTL register is set as 0. A new digest calculation is started when the START bit in the HAU_CTL register is 1, and SHA-1, SHA-224, SHA-256 and MD5 mode computation is chosen by the ALGM bits in the HAU_CTL register.

After a message block of 512 bit has been received through the HAU_DI register and the input FIFO, the processor starts the calculation with the information from DMA or the status of the CALEN bit.

The results can be finally read from the HAU_DO0...7 registers.

### 14.4.4. HMAC mode

HMAC mode is used for message authentication with a unique key chosen by the user. More information about the HMAC specifications please refer to “HMAC: keyed-hashing for message authentication, H. Krawczyk, M. Bellare, R. Canetti, February 1997”.

The HMAC algorithm can be represented as:

HMAC(input) = HASH[((key | opad) XOR 0x5c) | HASH(((key | ipad) XOR 0x36) | input)]

where ipad and opad are used to extend the key to 512 bits with several ‘0’ and ‘1’ is the concatenation operator.

There are four different phases in the HMAC mode:

1. Configure the HMS bit in the HAU_CTL register as 1 and set the ALGM bits for the desired algorithm. If the key size is longer than 64 bytes, then the KLM bit in the HAU_CTL register should also be set. After that, start the HAU core by setting the START bit.
2. The key is used as the input message to complete the calculation in HASH mode.
3. The new key used for the inner hush function is elaborated when the last word is accessed and computation has started.
4. After the first hash round, HAU core starts to receive the key for the outer hash function, usually, the outer hash function uses the same new key as the inner hash function. And when the last word of the key is entered and computation starts, the results are available in the HAU_DO registers.

## 14.5. HAU suspended mode

It is possible to suspend HASH or HMAC operation to perform a high-prior task first, then after

the high-prior task is finished, resume the suspended operation.

When suspending the current task, it is necessary to save the context of the current task from registers to memory, and then the task can be resumed by restoring the context from memory to the HAU registers.

The following steps can be performed to complete the HAU process of the suspended data blocks.

### 14.5.1. Transfer data by CPU

1. Stop the current data transmission and calculation. Wait for BUSY = 0, and wait for DIF = 1 when NWIF[3:0] is larger than 0 (do not wait for DIF = 1 when NWIF[3:0] is 0).Only when no data block is processing, users can save context.
2. Save the configuration. Save the content of HAU_INTEN, HAU_CFG, HAU_CTL, HAU_CTXS0 to HAU_CTXS37 (HAU_CTXS0 to HAU_CTXS53 when HMAC operation is processing) registers to memory.
3. Configure and process the new message.
4. Restore the process before. Restore the content from memory to HAU_INTEN, HAU_CFG and HAU_CTL registers.
5. Resume the message calculation. Set START bit of HAU_CTL register to 1, to restart a new message digest calculation.
6. Resume the previous core state. Restore the content from memory to HAU_CTXS0 ~ HAU_CTXS37 (HAU_CTXS0 ~ HAU_CTXS53 when HMAC operation is to be resumed) registers.
7. Continue the operation from where it suspended before.

### 14.5.2. Transfer data by DMA

1. Wait for BUSY = 0, then if the CCF bit of HAU_STAT register is set, the proceeding context switch is no longer need, otherwise wait for BUSY = 1 again.
2. Stop the current data transfer. Disable DMA data transmission, then clear DMAE bit of HAU_CTL register to disable DMA request.
3. Save the current configuration. Wait for BUSY = 0, then if the CCF bit of HAU_STAT register is set, the proceeding context switch is no longer need, otherwise save the content of HAU_INTEN, HAU_CFG, HAU_CTL, HAU_CTXS0 to HAU_CTXS37 (HAU_CTXS0 to HAU_CTXS53 when HMAC operation is processing) registers to memory.
4. Configure and process the new message.
5. Restore the process before. Restore the content from memory to HAU_INTEN, HAU_CFG and HAU_CTL registers.
6. Resume DMA channel transmission. Reconfigure the DMA channel to transfer data.
7. Resume the message calculation. Set START bit of HAU_CTL register to 1, to restart a new message digest calculation.
8. Resume the previous core state. Restore the content from memory to HAU_CTXS0 ~

HAU_CTXS37 (HAU_CTXS0 ~ HAU_CTXS53 when HMAC operation is to be resumed) registers.

9. Set DMAE bit of HAU_CTL register to 1, continue the operation from where it suspended before.

**Note:** If the value of NWIF[3:0] bits of HAU_CTL register is 0, it means the context switch occurs between two data blocks, at the time when the previous block is completely processed, and the next block has not been pushed into input FIFO, so there is no need to save and restore HAU_CTXS22 ~ HAU_CTXS37 registers.

## 14.6. HAU interrupt

There are two types of interrupt registers in HAU, which are both in HAU_STAT register. In HAU, the interrupt is used to indicate the situation of the input FIFO and the status of whether the digest calculation is completed.

Any of interrupts can be enabled or disabled by configuring the HAU interrupt enable register HAU_INTEN. Value 1 of the register bits enable the interrupts.

### 14.6.1. Input FIFO interrupt

When the processing of data pushed in the input FIFO is completed, then DIF is asserted. If input FIFO interrupt is enabled, when DIF is asserted, input FIFO interrupt will be asserted.

### 14.6.2. Calculation completion interrupt

When the digest calculation is finished, then CCF is asserted. If calculation completion interrupt is enabled, when CCF is asserted, calculation completion interrupt will be asserted.

## 14.7. Register definition

HAU base address: 0x4802 1400

### 14.7.1. Control register (HAU_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | ALGM[1] | Reserved | KLM |
| | | | | | | | | | | | | | rw | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | MDS | DINE | NWIF[3:0] | | | | ALGM[0] | HMS | DATAM[1:0] | | DMAE | START | Reserved | |
| | | rw | r | r | | | | rw | rw | rw | | rw | w | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18 | ALGM[1] | Algorithm selection bit 1 |
| 17 | Reserved | Must be kept at reset value. |
| 16 | KLM | Key length mode<br>0: Key length ≤ 64 bytes<br>1: Key length > 64 bytes<br>**Note:** This bit must be changed when no computation is processing. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | MDS | Multiple DMA Selection<br>Set this bit if hash message is large files and multiple DMA transfers are needed.<br>0: Single DMA transfers needed and CALEN bit is automatically set at the end of a DMA transfer<br>1: Multiple DMA transfers needed and CALEN bit is not automatically set at the end of a DMA transfer |
| 12 | DINE | DI register not empty<br>0: The DI register is empty<br>1: The DI register is not empty<br>**Note:** This bit is cleared when START bit or CALEN bit is set as 1. |
| 11:8 | NWIF[3:0] | Number of words in the input FIFO<br>**Note:** These bits are cleared when START bit set or a digest calculation starts (CALEN bit is set as 1, or DMA end of transfer) |
| 7 | ALGM[0] | Algorithm selection bit 0 |

| | | |
|---|---|---|
| | | This bit and bit 18 of CTL are written by software to select the SHA-1, SHA-224, SHA256 or the MD5 algorithm:<br>00: Select SHA-1 algorithm<br>01: Select MD5 algorithm<br>10: Select SHA224 algorithm<br>11: Select SHA256 algorithm |
| 6 | HMS | HAU mode selection, must be changed when no computation is processing<br>0: HASH mode selected<br>1: HMAC mode selected. If the key length is longer than 64 bytes, then KLM bit must also be set |
| 5:4 | DATAM[1:0] | Data type mode<br>Defines the format of the data entered into the HAU_DI register:<br>00: No swapping. The data written to HAU_DI is direct write to FIFO without swapping.<br>01: Half-word swapping. The data written into HAU_DI need half-word swapping before write to FIFO.<br>10: Byte swapping. The data written into HAU_DI need byte swapping before write to FIFO.<br>11: Bit swapping. The data written into HAU_DI need bit swapping before write to FIFO. |
| 3 | DMAE | DMA enable<br>0: DMA disabled<br>1: DMA enabled<br>**Note:** 1. This bit is cleared when transferring the last data of the message, but not cleared because of START.<br>2. When DMA is transferring, writing 0 to this bit will not stop the current transfer until the transfer is completed or START is set as 1. |
| 2 | START | Start the digest calculation<br>1: Start the digest of a new message<br>0: No effect<br>**Note:** Reading this bit always returns 0. |
| 1:0 | Reserved | Must be kept at reset value. |

### 14.7.2. Data input register (HAU_DI)

Address offset: 0x04
Reset value: 0x0000 0000

The data input register is used to transfer message with 512-bit blocks into the input FIFO for processing. Any new write operation to this register will be extended while the digest calculation is in process until it has been finished.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DI[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DI[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DI[31:0] | Message data input<br>When write to these registers, the current content pushed to IN FIFO and new value updates. When read, returns the current content. |

### 14.7.3. Configuration register (HAU_CFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CALEN | Reserved | | | VBL[4:0] | | | | |
| | | | | | | | w | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | CALEN | Digest calculation enable<br>0: No calculation<br>1: Start data padding with VBL prepared previously. Start the calculation of the last digest<br>**Note:** Reading this bit always returns 0. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | VBL[4:0] | Valid bits length in the last word<br>0x00: All 32 bits of the last data written to HAU_DI after data swapping are valid<br>0x01: Only bit [31] of the last data written to HAU_DI after data swapping are valid<br>0x02: Only bits [31:30] of the last data written to HAU_DI after data swapping are valid<br>0x03: Only bits [31:29] of the last data written to HAU_DI after data swapping are valid<br>...<br>0x1F: Only bits [31:1] of the last data written to HAU_DI after data swapping are valid |

**Note:** These bits must be configured before setting the CALEN bit.

### 14.7.4. Data output register (HAU_DO0..7)

Address offset: 0x0C
Reset value: 0x0000 0000

The data output registers are read only registers. They are used to receive results from the output FIFO. And they are reset by the START bit. Any read access when calculating will be extended until the calculation is completed.

In SHA-1 mode, HAU_DO0…4 are used.

In MD5 mode, HAU_DO0…3 are used.

In SHA-224 mode, HAU_DO0…6 are used.

In SHA-256 mode, HAU_DO0…7 are used.

#### HAU_DO0

Address offset: 0x0C and 0x310
Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO0[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO0[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

#### HAU_DO1

Address offset: 0x10 and 0x314
Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO1[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO1[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

#### HAU_DO2

Address offset: 0x14 and 0x318
Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO2[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO2[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

### HAU_DO3

Address offset: 0x18 and 0x31C

Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO3[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO3[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

### HAU_DO4

Address offset: 0x1C and 0x320

Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO4[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO4[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

### HAU_DO5

Address offset: 0x324

Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO5[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO5[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

### HAU_DO6

Address offset: 0x328

Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO6[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO6[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

### HAU_DO7

Address offset: 0x32C

Reset value: 0x0000 0000

These registers have to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO7[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DO7[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DO0..7[31:0] | Message digest result of hash algorithm |

## 14.7.5. Interrupt enable register (HAU_INTEN)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | CCIE | DIIE |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | CCIE | Calculation completion interrupt enable<br>0: Calculation completion interrupt is disabled |

| | | |
|---|---|---|
| | | 1: Calculation completion interrupt is enabled |
| 0 | DIIE | Data input interrupt enable<br>0: Data input interrupt is disabled<br>1: Data input interrupt is enabled |

### 14.7.6. Status and flag register (HAU_STAT)

Address offset: 0x24

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | BUSY | DMAS | CCF | DIF |
| | | | | | | | | | | | | r | r | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | BUSY | Busy bit<br>0: No processing<br>1: Data block is in process |
| 2 | DMAS | DMA status<br>0: DMA is disabled (DMAE = 0) and no transfer is processing<br>1: DMA is enabled (DMAE = 1) or a transfer is processing |
| 1 | CCF | Digest calculation completion flag<br>0: Digest calculation is not completed<br>1: Digest calculation is completed |
| 0 | DIF | Data input flag<br>0: A data is written to data input register<br>1: A data processing is completed (only the data in input FIFO will be processed) |

### 14.7.7. Context switch register x (HAU_CTXSx) (x = 0...53)

Address offset: 0xF8 + 0x04 × x, (x = 0...53)

Reset value: 0x0000 0002 when x = 0, 0x0000 0000 when x = 1 to 53.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTXx[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CTXx[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CTXx[31:0] | The complete internal status of the HAU core. Read and save the register data when a high-priority task is coming to be processed, and restore the saved data back to the registers to resume the suspended processing. |

# 15. Trigonometric Math Unit (TMU)

## 15.1. Overview

The Trigonometric Math Unit (TMU) is a fully configurable block that execute common trigonometric and arithmetic operations. The TMU can reduce the burden of CPU, and it is usually used in motor control, signal processing and many other applications.

The TMU can be used to calculate total 10 kinds of functions. The input / output data meet q1.31 or q1.15 fixed point format.

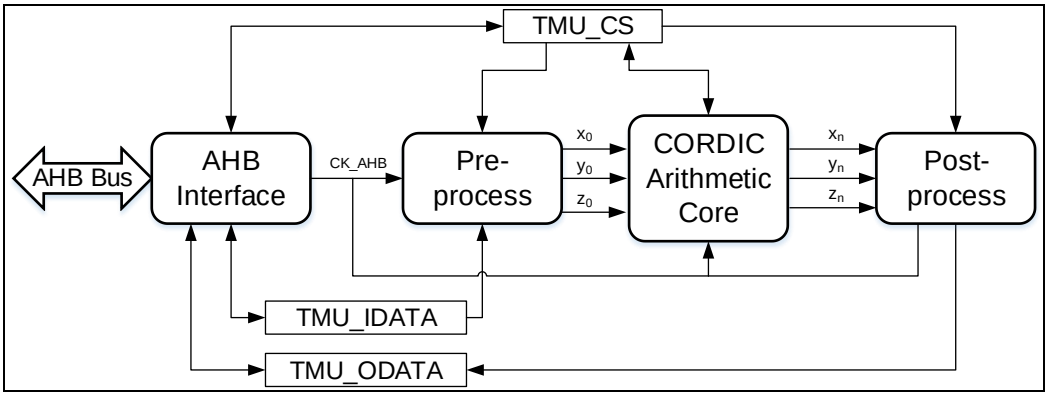
## 15.2. Characteristics

- 10 kinds of functions.
- Interrupt and DMA requests.
- The fixed point format is configurable.
- Programmable precision.
- CORDIC-algorithm core: circular system and hyperbolic system, rotation pattern and vectoring pattern.

## 15.3. Block diagram

*Figure 15-1. TMU block diagram* provides details of the internal configuration of the TMU.

**Figure 15-1. TMU block diagram**




The Pre-process module converts the contents in the TMU_IDATA register to obtain the initial data $(x_0,y_0,z_0)$ required by the CORDIC-arithmetic core. The contents of the TMU_IDATA register are in the format q1.31 or q1.15.

After the initial data $(x_0,y_0,z_0)$ is input to the CORDIC-algorithm core, it is iterated and calculated to obtain the $(x_n,y_n,z_n)$. The CORDIC-algorithm core supports circular system and hyperbolic system, and each system supports rotation pattern and vectoring pattern.

The Post-process module converts and scales the data $(x_n,y_n,z_n)$ and writes the processed results into TMU_ODATA register. The contents of the TMU_ODATA register are in q1.31 or q1.15 format.

## 15.4. Function overview

### 15.4.1. Data format and configuration

The input and output data of TMU module are fixed point signed integer format (q1.31 and q1.15 format).

In q1.31 format, the 31 bit is sign bit and 0~30 bits are fractional bits. The value range is [-1,1-$2^{-31}$], corresponding to [0x80000000,0x7FFFFFFF].

In q1.15 format, the 15 bit is sign bit and 0~14 bits are fractional bits. The value range is [-1,1-$2^{-15}$], corresponding to [0x8000,0x7FFF].

The IWITH bit in TMU_CS register is used to configure the fixed point format of the input data. Some modes (for example mode 0,m*sin(θ)) require two input datas, while some modes (for example mode 5, cosh(x)) require only one input data. The INUM bit in TMU_CS register is used to configure the number of input data. Detailed configuration refer to ***Table 15-1. Input data configuration***.

**Note:** When the input data is configured in q1.15 format, the TMU_IDATA register only needs to be written once, the first input data in the low half word, the second input data in the high half word. If the mode only needs one input data, only the low half word is used, and the high half word is not used.

**Table 15-1. Input data configuration**

| IWIDTH bit | INUM bit | Fixed format | Write operation to TMU_IDATA |
|---|---|---|---|
| 0 | 0 | q1.31 | Only one write operation |
| 0 | 1 | q1.31 | Two successive write operation |
| 1 | 0 | q1.15 | Only one write operation |
| 1 | 1 | q1.15 | Not available |

The OWITH bit in TMU_CS register is used to configure the fixed point format of the output data. Some modes (for example mode 0,m*sin(θ)) have two output datas, while some modes (for example mode 8, ln(x)) have only one output data. The ONUM bit in TMU_CS register is used to configure the number of output data. Detailed configuration refer to ***Table 15-2. Output data configuration***.

**Note:** When the output data is configured in q1.15 format, the TMU_IDATA register only needs to be read once, the first output data in the low half word, the second output data in the high half word. If the mode only needs one output data, only the low half word is used, and the high half word is not used.

**Table 15-2. Output data configuration**

| OWIDTH bit | ONUM bit | Fixed format | read operation to TMU_ODATA |
|---|---|---|---|
| 0 | 0 | q1.31 | Only one read operation |
| 0 | 1 | q1.31 | Two successive read operation |
| 1 | 0 | q1.15 | Only one read operation |
| 1 | 1 | q1.15 | Not available |

### 15.4.2. Mode configuration

The MODE[3:0] bit-field in TMU_CS register is used to configure the mode of the CORDIC-algorithm core. Different modes use different systems (circular or hyperbolic) and different patterns (rotation or vectoring). Detailed configuration refer to ***Table 15-3. TMU mode configuration***. Since the input and output data are in q1.31 or q1.15 format, some modes need to scale the actual input parameters. The FACTOR [2:0] bit-field in the TMU_CS register is used to configure the scaling factor.

**Table 15-3. TMU mode configuration**

| Mode | The first input data | The second input data | The first output data | The second output data | System and Pattern |
|---|---|---|---|---|---|
| Mode 0 | θ | m | $m*\cos(\theta)$ | $m*\sin(\theta)$ | Circular, Rotation |
| Mode 1 | θ | m | $m*\sin(\theta)$ | $m*\cos(\theta)$ | Circular, Rotation |
| Mode 2 | x | y | $\text{atan2}(y,x)$ | $\sqrt{x^2+y^2}$ | Circular, Vectoring |
| Mode 3 | x | y | $\sqrt{x^2+y^2}$ | $\text{atan2}(y,x)$ | Circular, Vectoring |
| Mode 4 | x | None | $\tan^{-1}(x)$ | None | Circular, Vectoring |
| Mode 5 | x | None | $\cosh(x)$ | $\sinh(x)$ | Hyperbolic, Rotation |
| Mode 6 | x | None | $\sinh(x)$ | $\cosh(x)$ | Hyperbolic, Rotation |
| Mode 7 | x | None | $\tanh^{-1}(x)$ | None | Hyperbolic, Vectoring |
| Mode 8 | x | None | $\ln(x)$ | None | Hyperbolic, Vectoring |
| Mode 9 | x | None | $\sqrt{x}$ | None | Hyperbolic, Vectoring |

Although TMU algorithm can only calculate a small number of functions directly, more functions can be obtained indirectly. For example, $e^x = \sinh(x) + \cosh(x)$.

#### Mode 0: m*cos(θ)

Mode 0 calculates the cosine of an angle. This mode take two input datas and generate two ouput datas. Detailed information refer to ***Table 15-4. Mode 0 description***.

**Table 15-4. Mode 0 description**

| Parameter | Range | Description |
|---|---|---|
| **First input data** | $\frac{\theta}{\pi} \in [-1,1)$ | The angle θ in radians range from –π to π. The θ must be divide by π in software to convert it to the |

| Parameter | Range | Description |
|---|---|---|
| | | range [-1,1), and then it is written to TMU_IDATA register according to the format of q1.31 or q1.15. |
| **Second input data** | m∈[0,1) | If 0≤m<1, it is written to TMU_IDATA register according to the format of q1.31 or q1.15. if m≥1, a scaling must be applied in software to convert it to the range [-1,1), and then it is written to TMU_IDATA register according to the format of q1.31 or q1.15. |
| **First output data** | m* cos(θ) ∈[-1,1) | If the previous software has shrunk m, the output data needs to be scaled up to obtain the real result. |
| **Second output data** | m* sin(θ) ∈[-1,1) | If the previous software has shrunk m, the output data needs to be scaled up to obtain the real result. |
| **FACTOR[2:0]** | Not available | keep at reset value 3'b000. |

**Note:** If $m > 1$, the scale is optional.

For example, calculating $100 * \cos\left(\frac{\pi}{2}\right)$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the first input parameter $\frac{\pi}{2}$. Angle $\frac{\pi}{2}$ is divided by π: $\frac{\frac{\pi}{2}}{\pi}$=0.5, 0.5 is 0x4000 in q1.15 format.
2. Software processes the second input parameter m. Modulus 100 is divided by 128: $\frac{100}{128}$=0.78125. 0.78125 is 0x6400 in q1.15 format.
3. The first input data 0x4000 is written into TMU_IDATA.
4. The second input data 0x6400 is written into TMU_IDATA. Then the TMU calculation starts.
5. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the first output data: $y_1 = \frac{100}{128} * \cos\left(\frac{\pi}{2}\right)$, reading the TMU_ODATA register again can get the second output data: $y_2 = \frac{100}{128} * \sin\left(\frac{\pi}{2}\right)$. The output data is q1.15 format.
6. Software processes the result. Since the previous software has shrunk m by 128, the output data needs to be scaled up by 128 to obtain the real result: $100 * \cos\left(\frac{\pi}{2}\right) = 128 * y_1$.

The scaling 128 is used for the input and output data in this example. Of course, other scaling, such as 101, can also be used.

### Mode 1: m*sin(θ)

Mode 1 calculates the sine of an angle. This mode take two input datas and generate two ouput datas. Detailed information refer to ***Table 15-5. Mode 1 description***.

**Table 15-5. Mode 1 description**

| Parameter | Range | Description |
|---|---|---|
| **First input data** | $\frac{\theta}{\pi} \in$ [-1,1) | The angle θ in radians range from $-\pi$ to $\pi$. The θ must be divide by $\pi$ in software to convert it to the range [-1,1), and then it is written to TMU_IDATA register according to the format of q1.31 or q1.15. |
| **Second input data** | m ∈ [0,1) | If 0≤m<1, it is written to TMU_IDATA register according to the format of q1.31 or q1.15. if m≥1, a scaling must be applied in software to convert it to the range [-1,1), and then it is written to TMU_IDATA register according to the format of q1.31 or q1.15. |
| **First output data** | m* sin(θ) ∈[-1,1) | If the previous software has shrunk m, the output data needs to be scaled up to obtain the real result. |
| **Second output data** | m* $\cos(\theta)$ ∈[-1,1) | If the previous software has shrunk m, the output data needs to be scaled up to obtain the real result. |
| **FACTOR[2:0]** | Not available | keep at reset value 3'b000. |

**Note:** If m > 1, the scale is optional.

For example, calculating $100*\sin\left(\frac{\pi}{2}\right)$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the first input parameter $\frac{\pi}{2}$. Angle $\frac{\pi}{2}$ is divided by π: $\frac{\frac{\pi}{2}}{\pi}$=0.5, 0.5 is 0x4000 in q1.15 format.
2. Software processes the second input parameter m. Modulus 100 is divided by 128: $\frac{100}{128}$=0.78125. 0.78125 is 0x6400 in q1.15 format.
3. The first input data 0x4000 is written into TMU_IDATA.
4. The second input data 0x6400 is written into TMU_IDATA. Then the TMU calculation starts.
5. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the first output data: $y_1=\frac{100}{128}*\sin\left(\frac{\pi}{2}\right)$, and reading the TMU_ODATA register again can get the second output data:$y_2=\frac{100}{128}*\cos\left(\frac{\pi}{2}\right)$. The output data is q1.15 format.
6. Software processes the result. Since the previous software has shrunk m by 128, the output data needs to be scaled up by 128 to obtain the real result: $100*\sin\left(\frac{\pi}{2}\right)=128*y_1$.

The scaling 128 is used for the input and output data in this example. Of course, other scaling, such as 101, can also be used.

### Mode 2: phase= atan2 (y,x)

Mode 2 calculates the atan2(y,x) of a vector (x,y). This mode take two input datas and generate two ouput datas. Detailed information refer to ***Table 15-6. Mode 2 description***.

**Table 15-6. Mode 2 description**

| Parameter | Range | Description |
|---|---|---|
| **First input data** | x∈[-1,1) | The abscissa value in Cartesian coordinate system. If x≥1 or x<-1, software scaling is required. |
| **Second input data** | y∈[-1,1) | The ordinate value in Cartesian coordinate system. If y≥1 or y<-1, software scaling is required. |
| **First output data** | θ∈[-1,1) | Angle, [-1,1) corresponding [-π,π). The output data is multiplied by π to get the real angle value. |
| **Second output data** | m∈[0,1) | Modulus, $m=\sqrt{x^2+y^2}$. If x and y have been scaled before, the modulus needs to be scaled equally. |
| **FACTOR[2:0]** | Not available | keep at reset value 3'b000. |

**Note:**

1. As long as one of x and y is out of the range [-1,1), x and y need to be scaled at the same scale at the same time, not only one. In this way, the angle corresponding to the coordinates before and after scaling can be kept same.
2. When $\sqrt{x^2+y^2}\geq 1$, the modulus m is only saturated to the maximum value of the fixed-point format. Before scaling x and y in the same scale, the scaling factor should be considered to avoid saturation of modulus.

For example, calculating θ=atan(5,80). The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameters (5,80). (5,80) is divided by 128. The result is (0.0390625,0.625). The q1.15 format is (0x0500,0x5000).
2. The first input data 0x0500 is written into TMU_IDATA.
3. The second input data 0x5000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the first output data θ, and reading the TMU_ODATA register again can get the second output data modulus m. The output data is q1.15 format.
5. Software processes the result. The first output data angle θ multiply π to get the real radian. Since the previous software has shrunk the input datas by 128, the second output data m needs to be scaled up by 128 to obtain the real modulus.

The scaling 128 is used for the input datas and the modulus in this example. Of course, other scaling, such as 81, can also be used.

### Mode 3: modulus=$\sqrt{x^2+y^2}$

Mode 3 calculates the modulus $\sqrt{x^2+y^2}$ of a vector (x,y). This mode take two input datas and generate two ouput datas. Detailed information refer to ***Table 15-7. Mode 3 description***.

**Table 15-7. Mode 3 description**

| Parameter | Range | Description |
|---|---|---|
| **First input data** | x∈[-1,1) | The abscissa value in Cartesian coordinate system. If x≥1 or x<-1, software scaling is required. |
| **Second input data** | y∈[-1,1) | The ordinate value in Cartesian coordinate system. If x≥1 or x<-1, software scaling is required. |
| **First output data** | m∈[0,1) | Modulus, m=$\sqrt{x^2+y^2}$. If x and y have been scaled before, the modulus needs to be scaled equally. |
| **Second output data** | θ∈[-1,1) | Angle, [-1,1) corresponding [-π,π). The output data is multiplied by π to get the real angle value. |
| **FACTOR[2:0]** | Not available | Keep at reset value 3'b000. |

**Note:**

1. As long as one of x and y is out of the range [-1,1), x and y need to be scaled at the same scale at the same time, not only one. In this way, the angle corresponding to the coordinates before and after scaling can be kept same.
2. When $\sqrt{x^2+y^2}$≥1, the modulus m is only saturated to the maximum value of the fixed-point format($1-2^{-15}$ or $1-2^{-31}$). Before scaling x and y in the same scale, the scaling factor should be considered to avoid saturation of modulus.

For example, calculating $\sqrt{5^2+80^2}$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameters (5,80). (5,80) is divided by 128. The result is (0.0390625,0.625). The q1.15 format is (0x0500,0x5000).
2. The first input data 0x0500 is written into TMU_IDATA.
3. The second input data 0x5000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the first output data modulus m, and reading the TMU_ODATA register again can get the second output data θ. The output data is q1.15 format.
5. Software processes the result. Since the previous software has shrunk the input datas by 128, the first output data m needs to be scaled up by 128 to obtain the real modulus. The second output data angle θ multiply π to get the real radian.

The scaling 128 is used for the input datas and the modulus in this example. Of course, other scaling, such as 81, can also be used.

### Mode 4: $\tan^{-1}(x)$

Mode 4 calculates the $\tan^{-1}(x)$. This mode take one input data and generate one ouput data. Detailed information refer to ***Table 15-8. Mode 4 description***.

**Table 15-8. Mode 4 description**

| Parameter | Range | Description |
|---|---|---|
| **Input data** | $\frac{x}{2^f} \in [-1,1)$ | If $x \in [-1,1)$, the software does not need to process it, and the scaling factor is $2^0$(FACTOR[2:0]=3'b000). If x is out of the range of [-1,1), it need to be scaled in software to ensure that $-1 \le \frac{x}{2^f} < 1$. Then write f to FACTOR[2:0] bit-field, and write the scaled data $\frac{x}{2^f}$ to TMU_IDATA in q1.15 or q1.31 format. |
| **Output data** | $\frac{\theta}{2^f} \in [-1,1)$ | Angle, [-1,1) corresponding [-π,π). The output data is multiplied by π and $2^f$ to get the real angle value $\theta$. |
| **FACTOR[2:0]** | $f \in [0,7]$ | The bit-field FACTOR[2:0] is configured as f |

For example, calculating $\tan^{-1}(100)$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameter 100. 100 is divided by 128 (f=7=3'b111). The result is 0.78125 and the q1.15 format is 0x6400.
2. The scaling factor f=3'b111 is written into FACTOR[2:0] bit-field in TMU_CS register.
3. The input data 0x6400 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the output data $\frac{\theta}{2^7}$. The output data is q1.15 format.
5. Software processes the result. The output data $\frac{\theta}{2^7}$ needs to be multiplied by π and $2^7$ to get the real radian.

### Mode 5: cosh (x)

Mode 5 calculates calculates the hyperbolic cosine of a hyperbolic angle x. This mode take one input data and generate two ouput datas. Detailed information refer to ***Table 15-9. Mode 5 description***.

**Table 15-9. Mode 5 description**

| Parameter | Range | Description |
|---|---|---|
| **Input data** | $\frac{x}{2} \in [-0.559,0.559]$ | $x \in [-1.118,1.118]$, a scaling factor $\frac{1}{2}$ is applied in softwate. Then write $\frac{x}{2}$ to TMU_IDATA in q1.15 or q1.31 format. |
| **First output data** | $\frac{\cosh(x)}{2} \in [0.5,0.846]$ | The output data is multiplied 2 to get the real hyperbolic cosine of a hyperbolic angle $x$. |

| Parameter | Range | Description |
|---|---|---|
| **Second output data** | $\frac{\sinh(x)}{2} \in [-0.683,0.683]$ | The output data is multiplied 2 to get the real hyperbolic sine of a hyperbolic angle $x$. |
| **FACTOR[2:0]** | 1 | The bit-field FACTOR[2:0] is configured as 3'b001 |

**Note:** The scaling factor FACTOR[2:0] must be 1.

For example, calculating $\cosh(1.0)$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameter 1.0. 1.0 is divided by 2 (f=3'b001). The result is 0.5 and the q1.15 format is 0x4000.
2. The scaling factor (f=3'b001) is written into FACTOR[2:0] bit-field in TMU_CS register.
3. The input data 0x4000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the first output data: $y_1 = \frac{\cosh 1.0)}{2}$, and reading the TMU_ODATA register again can get the second output data:$y_2 = \frac{\sinh(1.0)}{2}$. The output data is q1.15 format.
5. Software processes the result. The two output datas are multiplied 2 to get the real hyperbolic cosine and sine of a hyperbolic angle 1.0.

### Mode 6: sinh (x)

Mode 6 calculates calculates the hyperbolic sine of a hyperbolic angle $x$. This mode take one input data and generate two ouput datas. Detailed information refer to ***Table 15-10. Mode 6 description***.

**Table 15-10. Mode 6 description**

| Parameter | Range | Description |
|---|---|---|
| **Input data** | $\frac{x}{2} \in [-0.559,0.559]$ | x∈[-1.118,1.118], a scaling factor $\frac{1}{2}$ is applied in softwate. Then write $\frac{x}{2}$ to TMU_IDATA in q1.15 or q1.31 format. |
| **First output data** | $\frac{\sinh(x)}{2} \in [-0.683,0.683]$ | The output data is multiplied 2 to get the real hyperbolic sine of a hyperbolic angle $x$. |
| **Second output data** | $\frac{\cosh(x)}{2} \in [0.5,0.846]$ | The output data is multiplied 2 to get the real hyperbolic cosine of a hyperbolic angle $x$. |
| **FACTOR[2:0]** | 1 | The bit-field FACTOR[2:0] is configured as 3'b001 |

**Note:** The scaling factor FACTOR[2:0] must be 1.

For example, calculating $\sinh(1.0)$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameter 1.0. 1.0 is divided by 2 (f=3'b001). The result is 0.5 and the q1.15 format is 0x4000.
2. The scaling factor f=3'b001 is written into FACTOR[2:0] bit-field in TMU_CS register.

3. The input data 0x4000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the first output data: $y_1=\frac{\sinh(1.0)}{2}$, and reading the TMU_ODATA register again can get the second output data:$y_2=\frac{\cosh(1.0)}{2}$. The output data is q1.15 format.
5. Software processes the result. The two output datas are multiplied 2 to get the real hyperbolic cosine and sine of a hyperbolic angle 1.0.

### Mode 7: $\tanh^{-1}(x)$

Mode 7 calculates calculates the hyperbolic arctangent of a hyperbolic angle x. This mode take one input data and generate one ouput data. Detailed information refer to ***Table 15-11. Mode 7 description***.

**Table 15-11. Mode 7 description**

| Parameter | Range | Description |
|---|---|---|
| **Input data** | $\frac{x}{2}\in$[-0.403,0.403] | x∈[-0.806,0.806], a scaling factor $\frac{1}{2}$ is applied in softwate. Then write $\frac{x}{2}$ to TMU_IDATA in q1.15 or q1.31 format. |
| **Output data** | $\frac{\tanh^{-1}(x)}{2}\in[-0.559,0.559]$ | The output data is multiplied 2 to get the real hyperbolic arctangent of a hyperbolic angle $x$. |
| **FACTOR[2:0]** | 1 | The bit-field FACTOR[2:0] is configured as 3'b001 |

**Note**: The scaling factor FACTOR[2:0] must be 1.

For example, calculating $\tanh^{-1}(0.5)$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameter 0.5. 0.5 is divided by 2 (f=3'b001). The result is 0.25 and the q1.15 format is 0x2000.
2. The scaling factor f=3'b001 is written into FACTOR[2:0] bit-field in TMU_CS register.
3. The input data 0x2000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the output data: $y_1=\frac{\tanh^{-1}(0.5)}{2}$. The output data is q1.15 format.
5. Software processes the result. The output data is multiplied 2 to get the real hyperbolic arctangent of a hyperbolic angle 0.5.

### Mode 8: ln (x)

Mode 8 calculates calculates the natural logarithm of the input parameter x. This mode take one input data and generate one ouput data. Detailed information refer to ***Table 15-12. Mode 8 description***.

**Table 15-12. Mode 8 description**

| Parameter | Range | Description |
|---|---|---|
| **Input data** | $\frac{x}{2^f} \in [0.0535,0.875]$ | x∈[0.107,9.35], a scaling factor $2^{-f}$ is applied in softwate to ensure that $\frac{x}{2^f} < (1-\frac{1}{2^f})$. Then write $\frac{x}{2^f}$ to TMU_IDATA in q1.15 or q1.31 format. |
| **Output data** | $\frac{\ln(x)}{2^{(f+1)}} \in [-0.558,0.137]$ | The output data is multiplied $2^{(f+1)}$ to get the real ln (x). |
| **FACTOR[2:0]** | f∈[1,4] | The bit-field FACTOR[2:0] is configured as f |

For example, calculating ln (8). The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameter 8. 8 is divided by 16 (f=3'b100). The result is 0.5 and the q1.15 format is 0x4000.
2. The scaling factor f=3'b100 is written into FACTOR[2:0] bit-field in TMU_CS register.
3. The input data 0x4000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the output data: $y_1 = \frac{\ln(x)}{2^{(4+1)}}$. The output data is q1.15 format.
5. Software processes the result. The output data is multiplied $2^{(4+1)}$ to get the real the natural logarithm of 8.

In order to ensure the accuracy of calculation, it is recommended to use the scaling factors in ***Table 15-13. Recommended scaling factors in mode 8*** for different inputs.

**Table 15-13. Recommended scaling factors in mode 8**

| Input Parameter x range | FACTOR[2:0] | Input data x range |
|---|---|---|
| 0.107≤x<1 | 1 | [0.0535,0.5) |
| 1≤x<3 | 2 | [0.25,0.75) |
| 3≤x<7 | 3 | [0.375,0.875) |
| 7≤x<9.35 | 4 | [0.4375,0.584) |

### Mode 9: $\sqrt{x}$

Mode 9 calculates calculates the square root of the input parameter x. This mode take one input data and generate one ouput data. Detailed information refer to ***Table 15-14. Mode 9 description***.

**Table 15-14. Mode 9 description**

| Parameter | Range | Description |
|---|---|---|
| **Input data** | $\frac{x}{2^f} \in [0.027,0.875]$ | x∈[0.027,2.34], a scaling factor $2^{-f}$ is applied in softwate to ensure that $\frac{x}{2^f} < (1-\frac{1}{2^{f+2}})$. Then write $\frac{x}{2^f}$ |

| Parameter | Range | Description |
|---|---|---|
| | | to TMU_IDATA in q1.15 or q1.31 format. |
| **Output data** | $\frac{\sqrt{x}}{2^f} \in [0.04,1]$ | The output data is multiplied $2^f$ to get the real $\sqrt{x}$. |
| **FACTOR[2:0]** | $f \in [0,2]$ | The bit-field FACTOR[2:0] is configured as $f$ |

For example, calculating $\sqrt{2}$. The input and output are q1.15 format. The steps to calculate the function are as follows:

1. Software processes the input parameter $2$. 2 is divided by 4 ($f = 3'b010$). The result is 0.5 and the q1.15 format is 0x4000.
2. The scaling factor $f = 3'b010$ is written into FACTOR[2:0] bit-field in TMU_CS register.
3. The input data 0x4000 is written into TMU_IDATA. Then the TMU calculation starts.
4. When the ENDF flag is set to 1, reading the TMU_ODATA register can get the output data: $y_1 = \frac{\sqrt{2}}{2^2}$. The output data is q1.15 format.
5. Software processes the result. The output data is multiplied $2^2$ to get the real value of $\sqrt{2}$.

In order to ensure the accuracy of calculation, it is recommended to use the scaling factors in ***Table 15-15. Recommended scaling factors in mode 9*** for different inputs

**Table 15-15. Recommended scaling factors in mode 9**

| Input Parameter x range | FACTOR[2:0] | Input data x range |
|---|---|---|
| 0.027<x<0.75 | 0 | [0.027,0.75) |
| 0.75≤x<1.75 | 1 | [0.375,0.875) |
| 1.75≤x<2.341 | 2 | [0.4375,0.585) |

### 15.4.3. TMU precision

**Table 15-16. Precision vs. number of iterations**

| Mode | Number of iterations | Number of cycles | Max residual error(1) | |
|---|---|---|---|---|
| | | | q1.31 format | q1.15 format |
| Mode 0, Mode 1, Mode(2) 2, Mode(2) 3, Mode(4) 4 | 4 | 1 | $2^{-3}$ | $2^{-3}$ |
| | 8 | 2 | $2^{-7}$ | $2^{-7}$ |
| | 12 | 3 | $2^{-11}$ | $2^{-11}$ |
| | 16 | 4 | $2^{-15}$ | $2^{-15}$ |
| | 20 | 5 | $2^{-18}$ | $2^{-16}$ |
| | 24 | 6 | $2^{-19}$ | $2^{-16}$ |
| Mode 5, Mode 6, Mode 7, Mode(3) 8 | 4 | 1 | $2^{-2}$ | $2^{-2}$ |
| | 8 | 2 | $2^{-6}$ | $2^{-6}$ |
| | 12 | 3 | $2^{-10}$ | $2^{-10}$ |
| | 16 | 4 | $2^{-13}$ | $2^{-13}$ |
| | 20 | 5 | $2^{-17}$ | $2^{-15}$ |

| Mode | Number of iterations | Number of cycles | Max residual error(1) | |
|---|---|---|---|---|
| | | | q1.31 format | q1.15 format |
| | 24 | 6 | $2^{-18}$ | $2^{-15}$ |
| Mode(4) 9 | 4 | 1 | $2^{-7}$ | $2^{-7}$ |
| | 8 | 2 | $2^{-14}$ | $2^{-14}$ |
| | 12 | 3 | $2^{-19}$ | $2^{-15}$ |

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to $2^{-16}$ for q15 format or $2^{-20}$ for q31 format.
2. When the (x,y) are close to (0,0), the precision will sharply decrease.
3. FACTOR[2:0] = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. FACTOR[2:0] = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

### 15.4.4. TMU operation pending

If the TMU operation is ongoing, the further the contents written into TMU_IDATA register will be suspended. When the TMU operation completes (the result is read and the ENDF flag is cleared), if the number of suspended input data meets the configuration (defined in the suspended TMU_CS register), the TMU module will start a new TMU operation according to the pending configuration and data.

For example, if the configured TMU mode requires two 32-bit input data (IWIDTH = 0, INUM = 1), the TMU operation is started as sonn as two input data are written into the TMU_IDATA. If the second input data does not change in the next TMU operation, the INUM bit can be set to 0 at this time. After the previous TMU operation is completed, only one input data is written into TMU_IDATA. The second input data still uses the previous value in the later TMU operation as long as the TMU mode does not change.

**Note:** After a reset, the second input data is +1 (0x7FFFFFFF).

While an TMU operation is pending, the further contents written into TMU_IDATA or TMU_CS register will cover the original contents. The new TMU contents will be suspended, and the suspension of the original contents will be invalid.

### 15.4.5. Zero-overhead mode

After a TMU operation starts, the output data register can be read directly. And the bus will automatically insert the waiting cycle before the result is returned. The following steps can be followed:

1. Configure TMU_CS register as needed.

2. Start the TMU operation by written the input data into TMU_IDATA.
3. Configure the next TMU mode as required and write the input data into TMU_IDATA.
4. Read the result from the TMU_ODATA register. The waiting cycles are automatically inserted into the bus. After reading TMU_ODATA operation is completed, the suspended TMU operation will start automatically.
5. Go to step3.

### 15.4.6. Interrupt and DMA requests

When ENDF flag is set to 1, if the RIE bit in TMU_CS register is set to 1, an interrupt request is generated. After the ENDF flag is cleared to 0, the interrupt request is also cleared.

If the WDEN bit in TMU_CS register is set to 1 and no TMU operation is pending, DMA request is generated. The number of DMA request is depends on the INUM bit in TMU_CS register. If INUM is 0, only one DMA request is generated. If INUM is 1, two DMA requests are generated.

When ENDF flag is set to 1, if the RDEN bit in TMU_CS register is set to 1, DMA request is generated. The number of DMA request is depends on the ONUM bit in TMU_CS register. If ONUM is 0, only one DMA request is generated. If ONUM is 1, two DMA requests are generated.

## 15.5. Registers definition

TMU base address: 0x4001 0000

### 15.5.1. Control and status register (TMU_CS)

Address offset: 0x00

Reset value: 0x0000 0050

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ENDF | Reserved | | | | | | | | IWIDTH | OWIDTH | INUM | ONUM | WDEN | RDEN | RIE |
| r | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | FACTOR[2:0] | | | Reserved | | | | MODE[3:0] | | | |
| | | | | | rw | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | ENDF | End of TMU operation flag<br>0: No TMU operation or TMU operation is ongoing<br>1: TMU operation ends and the output data has been written into TMU_ODATA register.<br>This bit is set by hardware when TMU operation ends and the output data has been written into TMU_ODATA register.<br>This bit is cleared by reading the TMU_ODATA register (ONUM + 1) times.<br>**Note:** During this bit is set, no new TMU operation is started. |
| 30:23 | Reserved | Must be kept at reset value. |
| 22 | IWIDTH | Width of input data<br>0: 32-bit<br>1: 16-bit<br>This bit decide the data format for input data.<br>If 32-bit is configured, the input data in q1.31 format should be written into TMU_IDATA register.<br>If 16-bit is configured, the input data in q1.15 format should be written into TMU_IDATA register. The first input data is written into the lower half-word of TMU_IDATA register, and the second input data is written into the upper half-word of TMU_IDATA register. |
| 21 | OWIDTH | Width of output data<br>0: 32-bit<br>1: 16-bit<br>This bit decide the data format for the output data of TMU operation.<br>If 32-bit is configured, the TMU_ODATA contains the output data of TMU operation in q1.31 format. |

| | | |
|---|---|---|
| | | If 16-bit is configured, the TMU_ODATA contains the output data of TMU operation in q1.15 format. The the lower half-word of TMU_IDATA register contains the first output data, and the the upper half-word of TMU_IDATA register contains the second output data. |
| 20 | INUM | The number of times that the TMU_IDATA needs to be written<br>0: One 32-bit write operation. To start a new TMU operation, one 32-bit input data must written into TMU_IDATA register.<br>1: Two 32-bit write operation. To start a new TMU operation, two 32-bit input data must written into TMU_IDATA register.<br>**Note**: When the format of input data is q1.15 (IWIDTH=1) and the TMU mode need only one input data (INUM=0), the upper half-word of TMU_IDATA register is unused. |
| 19 | ONUM | The number of times that the TMU_ODATA needs to be read<br>0: One 32-bit read operation. When TMU operation completes, only one 32-bit result is transferred into TMU_ODATA register. Read the TMU_ODATA register once to clear ENDF flag.<br>1: Two 32-bit read operation. When TMU operation completes, two 32-bit results are transferred into TMU_ODATA register. Read the TMU_ODATA register twice to clear ENDF flag.<br>**Note**: When OWIDTH=1 (the format of output data is q1.15), only one 32-bit read is needed(ONUM=1). |
| 18 | WDEN | Enable DMA request to write TMU_IDATA<br>0: disabled. No DMA request is generated to write TMU_IDATA.<br>1: enabled. When no TMU operation is pending, the DMA request is generated. |
| 17 | RDEN | Enable DMA request to read TMU_ODATA<br>0: disabled. No DMA request is generated to read TMU_ODATA.<br>1: enabled. When ENDF is set, the DMA request is generated. |
| 16 | RIE | Enable interrupt request to read TMU_ODATA<br>0: disabled. No interrupt request is generated to read TMU_ODATA.<br>1: enabled. When ENDF is set, the interrupt request is generated. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:8 | FACTOR[2:0] | Scaling factor<br>This bit-field defines the scaling factor: $2^{FACTOR[2:0]}$.<br>000: $2^0$<br>001: $2^1$<br>010: $2^2$<br>…<br>110: $2^6$<br>111: $2^7$<br>When the actual input parameter exceeds the specified the input data range [-1,1), it is need to be divide by $2^{FACTOR[2:0]}$ and the output data is need to be multiplied by |

| | | |
|---|---|---|
| | | $2^{FACTOR[2:0]}$ to get the actual output result, details as follows:<br>TMU_IDATA = the actual input parameter/$2^{FACTOR[2:0]}$<br>the actual output result=TMU_ODATA*$2^{FACTOR[2:0]}$.<br>**Note:**<br>1. For mode8 and mode9, this bit field recommends some configurations for different parameters. For mode0, mode1, mode2 and mode3, this bit field is recommended to be configured as 3'b000. For mode5, mode6, and mode7, this bit field is recommended to be configured as 3'b001.<br>2. The input data(TMU_IDATA) and the output data(TMU_ODATA) are q1.31 or q1.15 format. |
| 7:4 | Reserved | Must be kept at reset value. |
| 3:0 | MODE[3:0] | Mode of TMU operation<br>0000: mode0,m* cos(θ)<br>0001: mode1, m*sin(θ)<br>0010: mode2, phase= atan2 (y,x)<br>0011: mode3, modulus=$\sqrt{x^2+y^2}$<br>0100: mode4, $\tan^{-1}$ (x)<br>0101: mode5, cosh (x)<br>0110: mode6, sinh (x)<br>0111: mode7, $\tanh^{-1}$ (x)<br>1000: mode8, ln (x)<br>1001: mode9, $\sqrt{x}$<br>1010~1111: reserved<br>**Note:**<br>x, θ: the first input data<br>y, m: the second input data |

## 15.5.2. Input data register (TMU_IDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDATA[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IDATA[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | IDATA[31:0] | The input data |

The input data is written into this register. For details, refer to ***Table 15-1. Input data configuration***.

**Note:**

1. When no TMU operation is is ongoing and the required number of arguments has been written, a new operation will be started automatically.
2. When the TMU operation is ongoing, the written data is suspended until the end of the TMU operation and the output data is read. During this period, if new input data is written, the previous suspension will be cancelled, and the new data will overwrite the previous data and be suspended.

### 15.5.3. Output data register (TMU_ODATA)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ODATA[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ODATA[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ODATA[31:0] | The output data<br>When TMU operation ends, the result is transferred into this register. For details, refer to ***Table 15-2. Output data configuration***.<br>**Note:**<br>1. When the ENDF bit is 1, the result of TMU operation can be obtained by reading the register.<br>2. When the read operation meeting the configuration is completed, the ENDF bit is cleared. |

# 16. Direct memory access controller (DMA)

## 16.1. Overview

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the MCU, thereby increasing system performance by off-loading the MCU from copying large amounts of data and avoiding frequent interrupts to serve peripherals needing more data or having available data.

Two AHB master interfaces and eight four-word depth 32-bit width FIFOs are presented in each DMA controller, which achieves a high DMA transmission performance. There are 16 independent channels in the DMA controller (8 for DMA0 and 8 for DMA1). Each channel is assigned a specific or multiple target peripheral devices for memory access request management. Two arbiters respectively for memory and peripheral are implemented inside to handle the priority among DMA requests.

Both the DMA controller and the Cortex®-M7 core implement data access through the system bus. An arbitration mechanism is implemented to solve the competition between these two masters. When the same peripheral is targeted, the MCU access will be suspended for some specific bus cycles. A round-robin scheduling algorithm is utilized in the bus matrix to guaranty at least half the bandwidth to the MCU.

## 16.2. Characteristics

- Two AHB master interface for transferring data, and one AHB slave interface for programming DMA.
- 16 channels (8 for DMA0 and 8 for DMA1) and each channel are configurable.
- Support independent single, 4, 8, 16-beat incrementing burst memory and peripheral transfer.
- Support switch-buffer transmission between peripheral and memory.
- Software DMA channel priority (low, medium, high, ultra high) and hardware DMA channel priority (DMA channel 0 has the highest priority and DMA channel 7 has the lowest priority).
- Support independent 8, 16, 32-bit memory and peripheral transfer.
- Support independent fixed and increasing address generation algorithm of memory and peripheral.
- Support circular transfer mode.
- Support three transfer modes:
  - Read from memory and write to peripheral.
  - Read from peripheral and write to memory.
  - Read from memory and write to memory.
- Support two data processing modes by use of the four-word depth 32-bit width FIFOs:
  - Multi-data mode: Pack/Unpack data when memory transfer width are different from

peripheral transfer width.

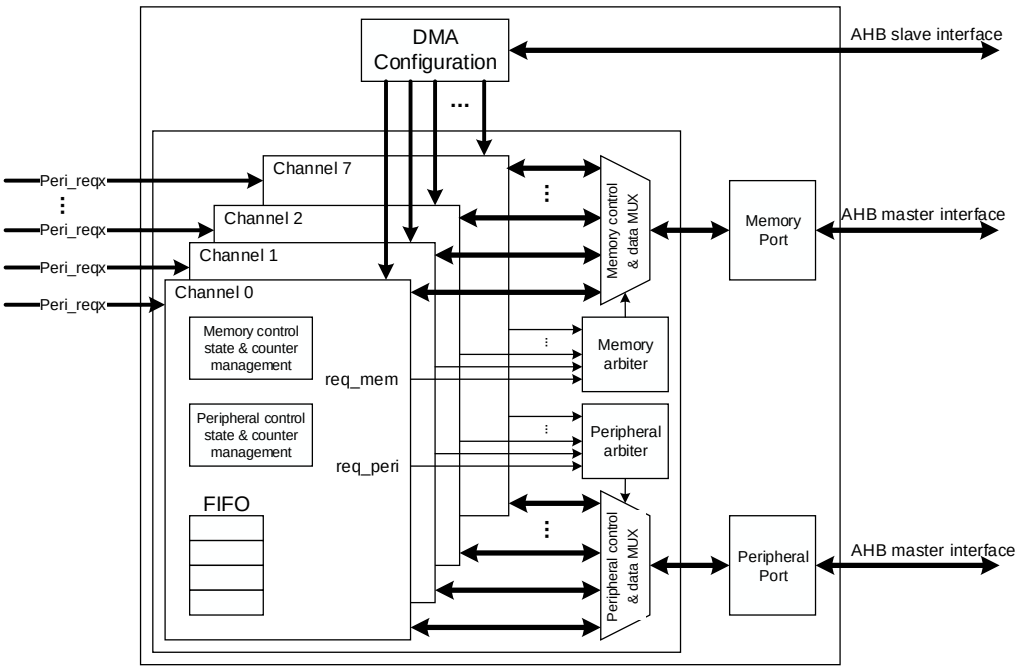
- Single-data mode: Read data from source when FIFO is empty and wirte data to destination when one data has been pushed into FIFO.

- One separate interrupt per channel with five types of event flags.
- Support interrupt enable and clear.

## 16.3. Function overview

### 16.3.1. Block diagram

**Figure 16-1. Block diagram of DMA**




As shown in ***Figure 16-1. Block diagram of DMA***, a DMA controller consists of four main parts:

- DMA configuration through AHB slave interface.
- Data access through two AHB master interfaces respectively for memory access and peripheral access.
- Two arbiters inside to manage multiple peripheral requests coming at the same time.
- Channel data management to control data packing/unpacking and counting.

The DMA controller transfers data from one address to another without CPU intervention. It supports multiple data sizes, burst types, address generation algorithm, priority levels and several transfer modes to allow for flexible application by configuring the corresponding bits in DMA registers. All the DMA registers can be 32-bit configured through AHB slave interface.

Three transfer modes are supported, including peripheral-to-memory, memory-to-peripheral and memory-to-memory, which is determined by the TM bits in the DMA_CHxCTL register,

as listed in ***Table 16-1. Transfer mode***.

**Table 16-1. Transfer mode**

| Transfer mode | TM[1:0] | Source | Destination |
|---|---|---|---|
| Peripheral to memory | 00 | DMA_CHxPADDR | DMA_CHxM0ADDR/ DMA_CHxM1ADDR |
| Memory to peripheral | 01 | DMA_CHxM0ADDR/ DMA_CHxM1ADDR | DMA_CHxPADDR |
| Memory to memory | 10 | DMA_CHxPADDR | DMA_CHxM0ADDR/ DMA_CHxM1ADDR |

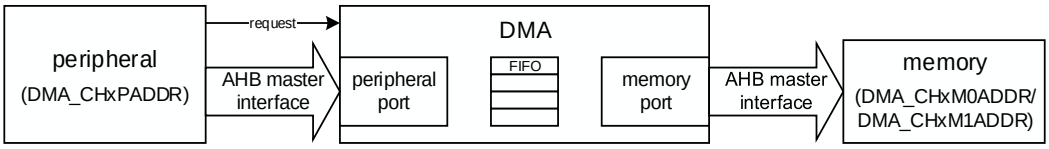
**Note:**

1. The MBS bit in DMA_CHxCTL register determines which is selected as the memory buffer address in DMA_CHxM0ADDR and DMA_CHxM1ADDR register. For more information, refer to section ***Switch-buffer mode***.

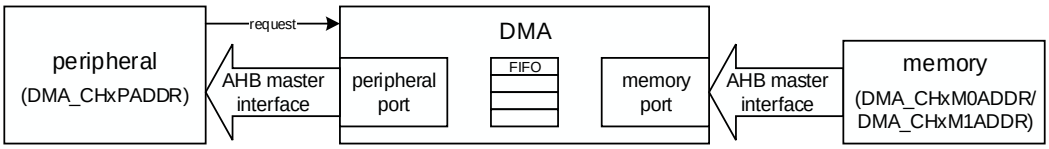
2. The TM bits in DMA_CHXCTL register are forbidden to configure to 0b11, or the channel will be automatically disabled.

**Figure 16-2. Data stream for three transfer modes**

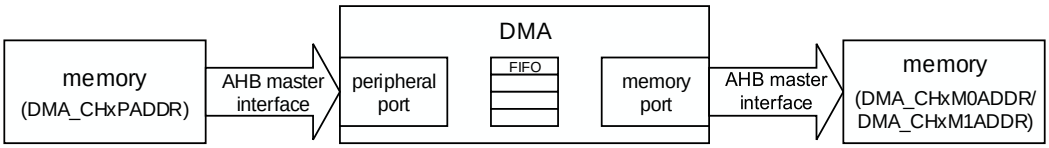
- peripheral-to-memory




- memory-to-peripheral




- memory-to-memory




As shown in ***Figure 16-2. Data stream for three transfer modes***, two AHB master interfaces are implemented in each DMA respectively for memory and peripheral.

- Memory to peripheral: read data from memory through AHB master interface for memory, and write data to peripheral through AHB master interface for peripheral.
- Peripheral to memory: read data from peripheral through AHB master interface for peripheral, and write data to memory through AHB master interface for memory.
- Memory to memory: read data from memory through AHB master interface for peripheral, and write data to another memory through AHB master interface for memory.

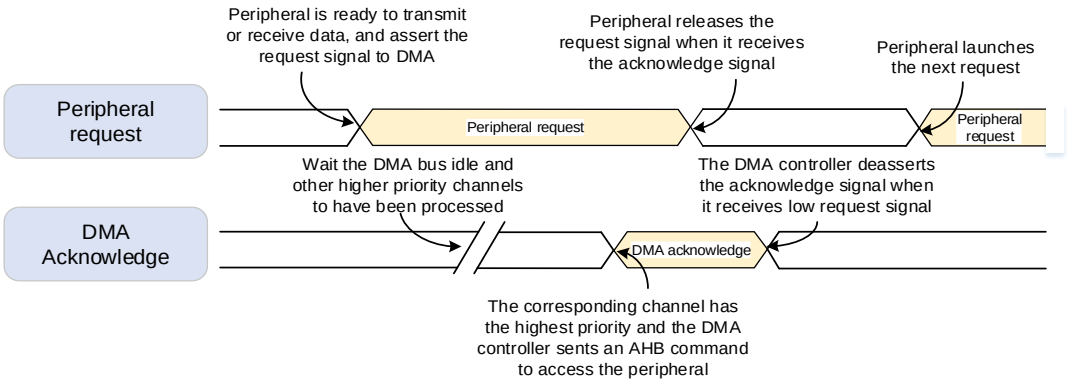
### 16.3.2. Peripheral handshake

To ensure a well-organized and efficient data transfer, a handshake mechanism is introduced between the DMA and peripherals, including a request signal and a acknowledge signal:

- Request signal asserted by peripheral to DMA controller, indicating that the peripheral is ready to transmit or receive data.
- Acknowledge signal responded by DMA to peripheral, indicating that the DMA controller has initiated an AHB command to access the peripheral.

*Figure 16-3. Handshake mechanism* shows how the handshake mechanism works between the DMA controller and peripherals.

**Figure 16-3. Handshake mechanism**




### 16.3.3. Data process

#### Arbitration

Two arbiters are implemented in each DMA respectively for memory and peripheral port. When two or more requests are received at the same time, the arbiter determines which channel is selected to respond according to the following priority rules:

- Software priority: Four levels, including low, medium, high and ultra high by configuring the PRIO bits in the DMA_CHxCTL register.
- Hardtware priority: For channels with equal software priority level, priority is given to the channel with lower channel number. For example, when channel 0 and channel 2 are configured with the same software priority, the priority of channel 0 is higher than that of channel 2.

#### Transfer width, burst and counter

##### Transfer width

PWIDTH and MWIDTH in the DMA_CHxCTL register indicate the data width of a peripheral and memory transfer seperately. The DMA supports 8-bit, 16-bit and 32-bit transfer width. In multi-data mode, if PWIDTH is not equal to MWIDTH, the DMA can automatically

packs/unpacks data to achieve an integrated and correct data transfer operation. In single-data mode, MWIDTH is automatically locked as PWIDTH by hardware immediately after enable the DMA channel.

**Transfer burst type**

PBURST and MBURST in the DMA_CHxCTL register indicate the burst type of a peripheral and memory transfer seperately. The DMA supports single burst, 4-beat, 8-beat and 16-beat incrementing burst for peripheral port and memory port. In single-data mode, only single burst type is supported and PBURST and MBURST are automatically locked as '00' by hardware immediately after enable the DMA channel.

In peripheral-to-memory or memory-to-peripheral mode, if PBURST is different from '00', DMA responses a increasing burst transfer of 4, 8, 16-beat based on the PBURST bits for each peripheral request. If the remaining bytes number of data item to be transferred is less than the bytes number needed for a burst transfer, the remaining data items are transferred in single transaction.

AMBA protocol specifies that bursts must not cross a 1KB address boundary, or a transfer error will be responsed to the master. In each DMA, the peripheral burst transfer crossing a 1KB address boundary is decomposed to 4, 8 or 16 single transactions depend on the PBURST bits, as the same as the memory burst transfer.

**Transfer counter**

The CNT bits in the DMA_CHxCNT register specifies how many data to be transmitted on the channel and must be configured before the CHEN bit is enabled. During the transmission, the CNT bits indicate the remaining number of data items to be transferred.

The CNT bits are related to peripheral transfer width, the number of data bytes to be transferred is the CNT bits multiplied by the byte number of the peripheral transfer width. For example, if the PWIDTH bits are equal to '10', and the number of data bytes to be transferred is CNT×4. The CNT bits is decreased by 1 when a single or a beat of the burst peripheral transfer (the source memory transfer in the memory-to-memory mode) has been completed even if the transfer mode is peripheral-to-memory or memory-to-memory.

When configuring the CNT bits, the following rules must be respected to guarantee a good DMA operation:

1. If the circular mode is disabled by clearing the CMEN bit in the DMA_CHxCTL register, the rules to configure the CNT bits in the DMA_CHxCNT register based on the transfer width are listed in the ***Table 16-2. CNT configuration***. The number of data bytes must be an integer multiple of the memory transfer width to guarantee an integrated single memory transfer.

**Note:** The number of data bytes does not need to be an interger multiple of the bytes number of a memory burst transfer or a peripheral burst number if the PBURST or/and MBURST bits are not equal to '00'. The remaining data that not enough for a burst transfer can be divided into single transfer automatically.

**Table 16-2. CNT configuration**

| PWIDTH | MWIDTH | CNT |
|---|---|---|
| 8-bit | 16-bit | Multiple of 2 |
| 8-bit | 32-bit | Multiple of 4 |
| 16-bit | 32-bit | Multiple of 2 |
| Others | | Any value |

1. If the circular mode is enabled by setting the CMEN bit in the DMA_CHxCTL register. The number of data bytes must be an integer multiple of the byte number of a peripheral burst transfer and a memory burst transfer to gurantee an integrated memory and peripheral burst transfer:

- $CNT / PBURST_beats$ must be an integer.
- $(CNT \times PWIDTH_bytes) / (MBURST_beats \times MWIDTH_bytes)$ must be an integer.
  - PWIDTH_bytes is the byte number of the peripheral transfer width, 1 for 8-bit, 2 for 16-bit and 4 for 32-bit.
  - PBURST_beats is the beat number of a peripheral burst transfer, 1 for single burst, 4 for INCR4 (4-beat incrementing burst), 8 for INCR8 (8-beat incrementing burst) and 16 for INCR16 (16-beat incrementing burst).
  - MWIDTH_bytes is the byte number of the peripheral transfer width, 1 for 8-bit, 2 for 16-bit and 4 for 32-bit.
  - MBURST_beats is the beat number of a peripheral burst transfer, 1 for single burst, 4 for INCR4, 8 for INCR8 and 16 for INCR16.

For example:

1. If PWIDTH is 16-bit, PBURST is INCR4, MWIDTH is 8-bit and MBURST is INCR16, $CNT/4$ and $(CNT\times2)/(1\times16)$ must be an integer, so the CNT bits must be configured to the multiple of 8.
2. If the If PWIDTH is 8-bit, PBURST is INCR16, MWIDTH is 16-bit and MBURST is INCR4, $CNT/16$ and $(CNT\times1)/(2\times4)$ must be an integer, so the CNT bits must be configured to the multiple of 16.

**Note:** When the switch-buffer mode is enabled by setting the SBMEN bit in the DMA_CHxCTL register, the circular mode is enabled automatically by hardware, and the above rules must also be respected.

### FIFO

A four-word depth FIFO is implemented as a data buffer for each DMA channel. Data reading from the source address is stored in the FIFO temporarily and transmitted to the destination through the destination port. Two data processing modes are supported depend on the FIFO configuration, including single-data mode and multi-data mode. When the transfer mode is memory-to-memory, only multi-data mode is supported to implement the DMA data processing.

**Multi-data mode**

The multi-data mode is selected by configuring the MDMEN bit in the DMA_CHxFCTL register to '1'.

In this mode, the DMA responds the source request when there is enough FIFO space for a source transfer, pushing the data reading from the source address into the FIFO. If the destination is a peripheral, the DMA responds the peripheral request when there is enough FIFO data for a peripheral burst transfer. If the memory is configured as the destination, the FIFO counter critical value configured in the FCCV bits of the DMA_CHxFCTL register controls the memory data processing. Only when the FIFO counter is reached the critical value, the data in the FIFO are entirely poped and written into the memory address.

To gurantee a good DMA behavior, the FIFO counter critical value (FCCV bits in the DMA_CHxFCTL register) must be an integer multiple of a memory burst transfer to ensure there is enough data for memory burst transfers. The configuration rules of the FIFO counter critical value depending on memory transfer width and memory burst types are listed in ***Table 16-3. FIFO counter critical value configuration rules***.

**Table 16-3. FIFO counter critical value configuration rules**

| MWIDTH | MBURST | FIFO counter critical value | | | |
|---|---|---|---|---|---|
| | | 1-word | 2-word | 3-word | 4-word |
| 8-bit | single | 4 single transactions | 8 single transactions | 12 single transactions | 16 single transactions |
| | INCR4 | 1 burst transaction | 2 burst transactions | 3 burst transactions | 4 burst transactions |
| | INCR8 | ERROR | 1 burst transaction | ERROR | 2 burst transactions |
| | INCR16 | ERROR | ERROR | ERROR | 1 burst transaction |
| 16-bit | single | 2 single transactions | 4 single transactions | 6 single transactions | 8 single transactions |
| | INCR4 | ERROR | 1 burst transaction | ERROR | 2 burst transactions |
| | INCR8 | ERROR | ERROR | ERROR | 1 burst transaction |
| | INCR16 | ERROR | ERROR | ERROR | ERROR |
| 32-bit | single | 1 single transaction | 2 single transactions | 3 single transactions | 4 single transactions |
| | INCR4 | ERROR | ERROR | ERROR | 1 burst transactions |
| | INCR8 | ERROR | ERROR | ERROR | ERROR |
| | INCR16 | ERROR | ERROR | ERROR | ERROR |

**Note:** When the transfer mode is peripheral-to-memory, if the PBURST_beats×PWIDTH_bytes = 16, the FIFO counter critical value must not be equal to '2b10'. When receiving a peripheral request, DMA initiates a peripheral burst transfer to entirely fill the FIFO. Then DMA lanches memory burst transfers to pop three words from the FIFO depending on the FIFO counter critical value and a word is still remained in the FIFO. There is no enough space for a peripheral burst transfer and the FIFO counter critical value is not reached, which makes DMA transfer frozen.

### Single-data mode

The single-data mode is selected by configuring the MDMEN bit in the DMA_CHxFCTL register to '0'. In this mode, only single transfer is supported to implement the DMA data access, and the FIFO counter critical value configured in the FCCV bits of the DMA_CHxFCTL register has no meaning.

In single-data mode, DMA responds the source request only when the FIFO is empty, pushing the data reading from the source address into the FIFO whatever the source transfer width is. When the FIFO is not empty, DMA responds the destination request, poping the data from the FIFO and writing it to the destination address.

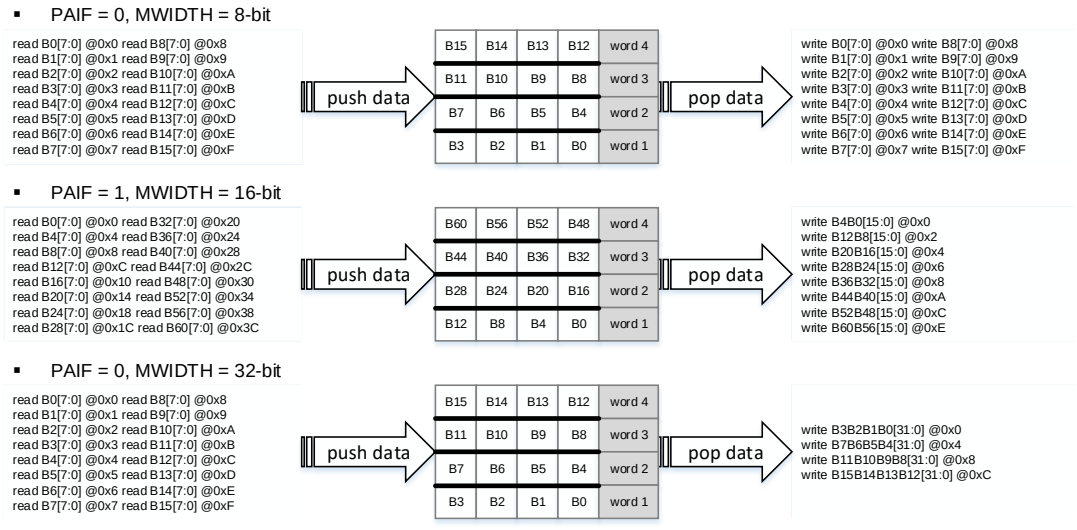
## Pack/Unpack

In single-data mode, the MWIDTH bits are equal to the PWIDTH bits by force, data packing/unpacking is not needed.

In multi-data mode, the independent PWIDTH and MWIDTH bits configuration are supported for flexible DMA transfer. When the PWIDTH bits and the MWIDTH bits are not equal, DMA reading access and writing access are executed in different transfer width, and DMA packs/unpacks the data automatically. In DMA transfer operation, only little-endian addressing for both memory and peripheral is supported.

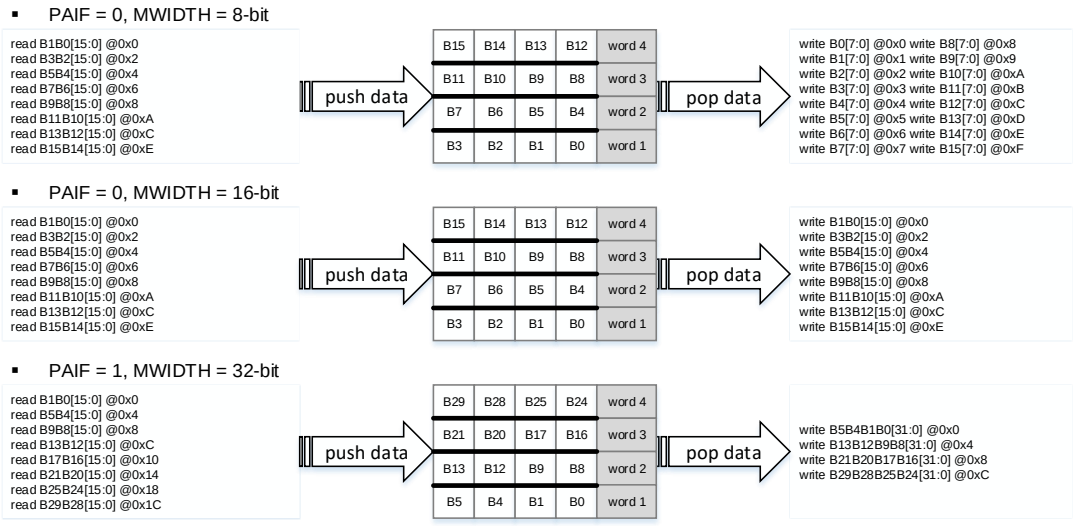
Suppose the CNT bits are 16, the PWIDTH bits are equal to '00', and both PNAGA and MNAGA are set. The DMA transfer operations for different MWIDTH are shown in the ***Figure 16-4. Data packing/unpacking when PWIDTH = '00'***.

**Figure 16-4. Data packing/unpacking when PWIDTH = '00'**



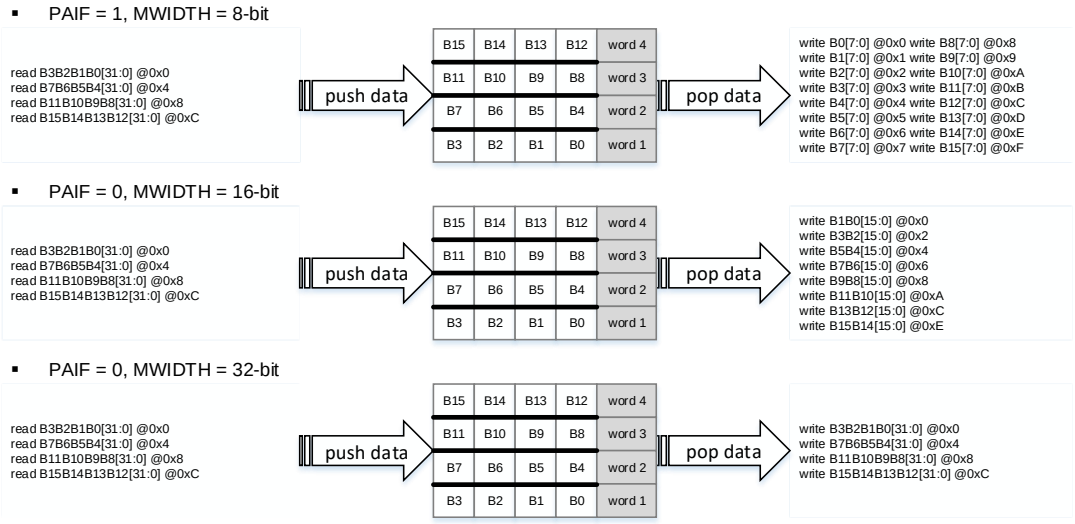

Suppose the CNT bits are 8, the PWIDTH bits are equal to '01', and both PNAGA and MNAGA are set. The DMA transfer operations for different MWIDTH are shown in the ***Figure 16-5. Data packing/unpacking when PWIDTH = '01'***.

**Figure 16-5. Data packing/unpacking when PWIDTH = ‘01’**




Suppose DMA_CHxCNT is 4, the PWIDTH bits are equal to ‘10’, and both PNAGA and MNAGA are set. The DMA transfer operations for different MWIDTH are shown in the ***Figure 16-6. Data packing/unpacking when PWIDTH = ‘10’***.

**Figure 16-6. Data packing/unpacking when PWIDTH = ‘10’**




## 16.3.4. Address generation

Two kinds of address generation algorithm are implemented independently for memory and peripheral, including the fixed mode and the increased mode. The PNAGA and MNAGA bit in the DMA_CHxCTL register are used to configure the next address generation algorithm of peripheral and memory.

In the fixed mode, the next address is always equal to the base address configured in the base address registers (DMA_CHxPADDR, DMA_CHxM0ADDR, and DMA_CHxM1ADDR).

In the increasing mode, the next address is euqal to the current address plus 1 or 2 or 4, depending on the transfer data width. In Multi-data mode with PBURST in the DMA_CHxCTL

register is ‘00’, if PAIF in the DMA_CHxCTL register is enabled, the next peripheral address increment is fixed to 4, and has nothing to do with the peripheral transfer data width. The PAIF has no meaning to the memory address generation.

**Note:** If PAIF in the DMA_CHxCTL register is enable, the peripheral base address configured in the DMA_CHxPADDR register must be 32-bit alignment.

### 16.3.5. Circular mode

Circular mode is implemented to handle continue peripheral requests. The CMEN bit in DMA_CHxCTL register is used to enable/disable the circular mode. Circular mode is available only when DMA controls the transfer flow.

In circular mode, the CNT bits are automatically reloaded with the pre-programmed value and the full transfer finish flag is asserted at the end of every DMA transfer. DMA can always respond the peripheral request until a transfer error is detected or the CHEN bit in the DMA_CHxCTL register is cleared.

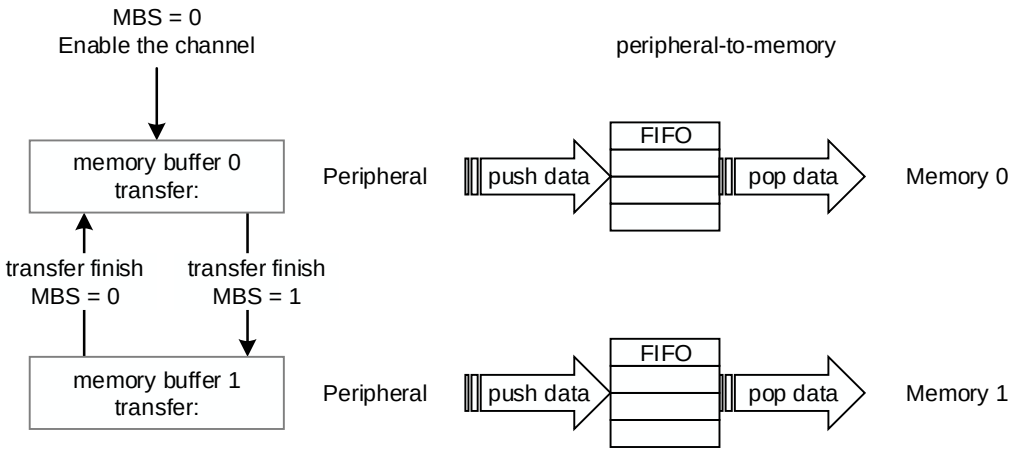
### 16.3.6. Switch-buffer mode

Similar to circular mode, switch-buffer mode is also implemented to handle continues peripheral requests. The SBMEN bit in the DMA_CHxCTL register is used to enable/disable the switch-buffer mode. When the switch-buffer mode is enabled, the circular mode is automatically enabled immediately after the channel is enabled. Switch-buffer mode is only available when the transfer mode is peripheral-to-memory or memory-to-peripheral. When the transfer mode is memory-to-memory, the switch-buffer mode is automatically disabled immediately after the channel is enabled.

Switch-buffer mode is supported with two memory buffers and the base address of the two memory buffers are separately configured in the DMA_CHxM0ADDR and DMA_CHxM1ADDR register. In switch-buffer mode, the DMA memory pointer switches from the current memory buffer to another at the end of every DMA transfer. During the DMA transmission, the memory buffer not being processed by DMA can be accessed by other AHB masters. In switch-buffer mode, the base address of the memory buffer not accessed by DMA can be updated even if the channel is enabled.

The MBS bit in the DMA_CHxCTL register is configured to select which memory buffer is accessed by DMA at the first DMA transfer before the channel is enabled. In switch-buffer mode, this bit switches automatically between ‘0’ and ‘1’ at the end of every DMA transfer, and can be used as a flag indicating the current memory buffer accessed by DMA during the transmission. The DMA operation of switch-buffer mode are shown in ***Figure 16-7. DMA operation of switch-buffer mode***.

**Figure 16-7. DMA operation of switch-buffer mode**




## 16.3.7. Transfer operation

Three transfer modes are supported to implement the data transfer, including peripheral-to-memory, memory-to-peripheral and memory-to-memory. Memory and peripheral can be configured as source and destination relatively.

**Memory transfer**

- Peripheral-to-memory mode:
  - In single-data mode, when the FIFO is not empty, DMA initiates a single memory transfer and writes data into the corresponding memory address.
  - In multi-data mode, when the FIFO counter reaches the critical value, DMA starts single or burst memory transfers to entirely fetch the FIFO data and write to the memory.
- Memory-to-peripheral mode:
  - In single-data mode, when the channel is enabled, DMA starts a single memory transfer and pushes the reading data into the FIFO immediately. During the transmission, the memory transfer is initiated only when the FIFO is empty.
  - In multi-data mode, when the channel is enabled, DMA starts several single or burst transfers to fill up the FIFO whether the peripheral request is asserted or not. During the transmission, the memory transfer is initiated once when there is enough space for it in the FIFO.
- Memory-to-memory mode: When the FIFO counter reaches the critical value, DMA starts single or burst memory transfers to entirely fetch the FIFO data and write to the memory.

**Peripheral transfer**

- Peripheral-to-memory mode: When receiving a peripheral request and there is enough space in the FIFO for a peripheral transfer, DMA starts a peripheral transfer and pushes the reading data into the FIFO.
- Memory-to-peripheral mode: When receiving a peripheral request and there is enough data in the FIFO for a peripheral transfer, DMA starts a peripheral transfers to fetch the

FIFO data and write to the peripheral.

### 16.3.8. Transfer finish

The DMA transfer is finished automatically and the FTFIFx bit in the DMA_INTF0 or DMA_INTF1 register is set when one of the following situations occurs:

- Transfer completion.
- Software clear.
- Error detection.

#### Transfer completion

When enabled, the DMA begins to transfer data between peripheral and memory. After the pre-programmed number of data items has been transferred successfully, the DMA transfer is completed and the CHEN bit is automatically cleared in the DMA_CHxCTL register.

- Peripheral-to-memory mode: When the CNT bits reach to zero and the contents of the FIFO have been entirely transferred into the memory, an end of transfer is generated.
- Memory-to-peripheral mode: When the CNT bits reach to zero, an end of transfer is achieved.
- Memory-to-memory: When the CNT bits reach to zero and the contents of the FIFO have been entirely transferred into the memory, an end of transfer is generated.

#### Software clear

The DMA transfer can be stopped by clearing the CHEN bit in the DMA_CHxCTL register by software. After the software cleared operation, the CHEN bit is still read as 1 to indicate that there are memory or peripheral transfers still active or the remaining data in the FIFO need to be transferred.

- Peripheral-to-memory: After the software cleared operation, the peripheral transfer is stopped when the current single or burst transfer is completed. To ensure that the data had been read from peripheral can be entirely transferred into the memory, the memory transfer continues to be active until the FIFO is empty. If the remaining byte number in the FIFO is not enough for a burst memory transfer, these data items are transferred in single transaction. If the remaining byte number is less than the memory transfer width, these data items are still written in memory transfer width with MSBs filled with zero. The software can read the CNT bits to calculate the number of valid data items in the memory. After the contents of the FIFO has been entirely transferred into the memory, the CHEN bit is cleared automatically by hardware and the FTFIFx bit in the DMA_INTF0 or DMA_INTF1 register is set.
- Memory-to-peripheral: After the software cleared operation, the DMA transfer is stopped when the current memory and peripheral transfer are completed. Then the CHEN bit is cleared automatically by hardware and the FTFIFx bit in the DMA_INTF0 or DMA_INTF1 register is set.
- Memory-to-memory: The same as the peripheral-to-memory mode with the source

memory transfer is implemented through the peripheral port.

#### Error detection

Three types error can disable the DMA transfer:

- FIFO error: When a wrong FIFO configuration is detected, the DMA channel is disabled immediately without starting any transfers. In this situation, the FTFIFx is not asserted. For more information about the FIFO error, refer to section ***Error***.
- Bus error: When the memory or peripheral port attempts to access an address beyond the access scope, a bus error is detected and the DMA transfer is stopped immediately without setting the FTFIFx. If this error is aroused by the peripheral port, the CNT bits are still decreased by 1. For more information about the bus error, refer to section ***Error***.
- Register access error: In switch-buffer mode, an access error is detected when a write command is active on the memory base address register which is being accessed by DMA. When this error occurs, the DMA operation is the same as it after the CHEN bit software cleared. For more information about the register access error, refer to section ***Error***.

### 16.3.9. Channel configuration

When starting a new DMA transfer, it is recommended to respect the following steps:

1. Read the CHEN bit and judge whether the channel is enabled or not. If the channel is enabled, clear the CHEN bit by software or wait the current DMA transfer finished. When the CHEN bit is read as '0', configuring and starting a new DMA transfer is allowed.
2. Clear the FTFIFx bit in the DMA_INTF0 or DMA_INTF1 register, or a new DMA transfer can not be re-enabled.
3. Configure the TM bits in the DMA_CHxCTL register to set the transfer mode.
4. Configure the memory and peripheral burst types, the target memory buffer, switch-buffer mode, priority of the channel, memory and peripheral transfer width, memory and peripheral address generation algorithm, circular mode in the DMA_CHxCTL register.
5. Configure multi-data mode, and the FCCV bits to set the FIFO counter critical value if multi-data mode is enabled in the DMA_CHxFCTL register.
6. Configure the enable bit for full transfer finish interrupt, half transfer finish interrupt, transfer access error interrupt, single-data mode exception interrupt in the DMA_CHxCTL register and the enable bit for FIFO error and exception interrupt in the DMA_CHxFCTL register.
7. Configure the DMA_CHxPADDR register for setting the peripheral base address.
8. If the switch-buffer mode is enabled, configure the memory base address in DMA_CHxM0ADDR and DMA_CHxM1ADDR register . If only one memory buffer is to be used, configure the DMA_CHxM0ADDR or DMA_CHxM1ADDR corresponding with

the MBS bit in the DMA_CHxCTL register.

9. Configure the DMA_CHxCNT register to set the total transfer data number.
10. Configure the CHEN bit with '1' in the DMA_CHxCTL register to enable the channel.

When restarting the suspended DMA transfer, it is recommended to respect the following steps:

1. Read the CHEN bit and ensure the DMA suspend operation has been completed. When the CHEN bit is read as '0', DMA is idle, restarting the DMA transfer is allowed.
2. Clear the FTFIFx bit in the DMA_INTF0 or DMA_INTF1 register, or the DMA transfer can not be re-enabled.
3. Read the DMA_CHxCNT register to obtain the number of the remaining data items and calculate the number of the data items had already been transferred.
4. Configure the DMA_CHxPADDR register to update the peripheral address pointer.
5. Configure the DMA_CHxM0ADDR or the DMA_CHxM1ADDR register to update the memory address pointer.
6. Configure the DMA_CHxCNT with the number of the remaining data items.
7. Configure the CHEN bit with '1' in the DMA_CHxCTL register to restart the channel.

## 16.4. Interrupts

Each DMA channel has a dedicated interrupt. There are five interrupt events connected to each interrupt, including full transfer finish interrupt, half transfer finish interrupt, transfer access error interrupt, single-data mode exception interrupt, and FIFO error and exception interrupt. A DMA channel interrupt may be produced when any interrupt event occurs on the channel.

Each interrupt event has a dedicated flag bit in the DMA_INTF0 or DMA_INTF1 register, a dedicated clear bit in the DMA_INTC0 and DMA_INTC1 register, and a dedicated enable bit in the DMA_CHxCTL and CHxFCTL register, as described in the ***Table 16-4. DMA interrupt events***.

**Table 16-4. DMA interrupt events**

| Interrupt event | Flag bit | Enable bit | Clear bit |
|---|---|---|---|
| | DMA_INTF0 or DMA_INTF1 | DMA_CHxCTL or DMA_CHxFCTL | DMA_INTC0 or DMA_INTC1 |
| Full transfer finish | FTFIF | FTFIE | FTFIFC |
| Half transfer finish | HTFIF | HTFIE | HTFIFC |
| Transfer access error | TAEIF | TAEIE | TAEIFC |
| Single-data mode exception | SDEIF | SDEIE | SDEIFC |

| Interrupt event | Flag bit | Enable bit | Clear bit |
|---|---|---|---|
| | DMA_INTF0 or DMA_INTF1 | DMA_CHxCTL or DMA_CHxFCTL | DMA_INTC0 or DMA_INTC1 |
| FIFO error and exception | FEEIF | FEEIE | FEEIFC |

These five events can be divided into three types:

- Flag: Full transfer finish flag and half transfer finish flag.
- Exception: Single-data mode exception and FIFO exception.
- Error: Transfer access error and FIFO error.

When the exception events occur, the DMA transmission is not affected and continues transferring normally. When the error events are detected, the DMA transmission is stopped. These three types of event are described in detail in the following sections.

### 16.4.1. Flag

Two flag events are supported, including full transfer finish flag and half transfer finish flag.

The full transfer finish flag is asserted, when one of the following situations occurs:

- The CNT bits reach to zero.
- When the channel is disabled by software before the end of the transfer, the current memory and peripheral is completed and the contents of the FIFO are entirely written into the memory in peripheral-to-memory or memory-to-memory mode.
- When the channel is disabled because of register access error before the end of the transfer, the current memory and peripheral is completed and the contents of the FIFO are entirely written into the memory in peripheral-to-memory or memory-to-memory mode.

When the full transfer finish flag is asserted and the enabled bit for the full transfer finish interrupt is set, an interrupt is generated.

The half transfer finish flag is asserted, only when DMA is the transfer flow controller and half of the CNT bits are transferred. If peripheral is the transfer flow controller, DMA does not know when half of data items has been transferred and the half transfer finish flag will stay zero.

When the half transfer finish flag is asserted and the enabled bit for the half transfer finish interrupt is set, an interrupt is generated.

### 16.4.2. Exception

Two exception events are supported, including single-data mode exception and FIFO exception. These exceptions have no effect on the DMA transmission.

#### Single-data mode exception

This exception can be detected only when the single-data mode is enabled and the transfer

mode is peripheral-to-memory. When a peripheral request is valid and the FIFO is not empty, there are two or more data items stored in the FIFO after responding the peripheral request, which could be a problem for the subsequent processing of the data and the single-data mode exception bit SDEIFx will be set.

When the single-data mode exception is asserted and the enabled bit for the single-data mode exception interrupt is set, an interrupt is generated.

#### FIFO exception

When a FIFO underrun or a FIFO overrun condition occurs, the FIFO exception is asserted. This exception can be detected only when the transmission is between peripheral and memory.

In peripheral-to-memory mode, when a peripheral request is valid and there is not enough space in the FIFO for the single or burst peripheral transfer, a FIFO overrun condition is detected. This peripheral request is not responded until the FIFO space is enough, and the accuracy of the data transmission will not be destroyed.

In memory-to-peripheral mode, when a peripheral request is valid and there is not enough data in the FIFO for the single or burst peripheral, a FIFO underrun condition is detected. This peripheral request is not responded until the data number in the FIFO is enough, and the accuracy of the data transmission will not be destroyed.

When the FIFO exception is asserted and the enabled bit for the FIFO error and exception interrupt is set, an interrupt is generated.

### 16.4.3. Error

FIFO error and transfer access error (including the register access error and bus error) can be detected during the DMA transmission, and the transmission can be stopped when one of the errors occurs.

#### FIFO error

For a DMA operation, when the multi-data mode is enabled, the right and wrong configurations of the FIFO counter critical value corresponding with the memory transfer width and memory burst types are listed in ***Table 16-3. FIFO counter critical value configuration rules***.

If a wrong configuration is detected after enable the channel, a FIFO error is generated and the channel is disabled immediately without starting any transfers.

When the FIFO error is asserted and the enabled bit for the FIFO error and exception interrupt is set, an interrupt is generated.

#### Register access error

The register access error is detected only when the switch-buffer is enabled. If the software

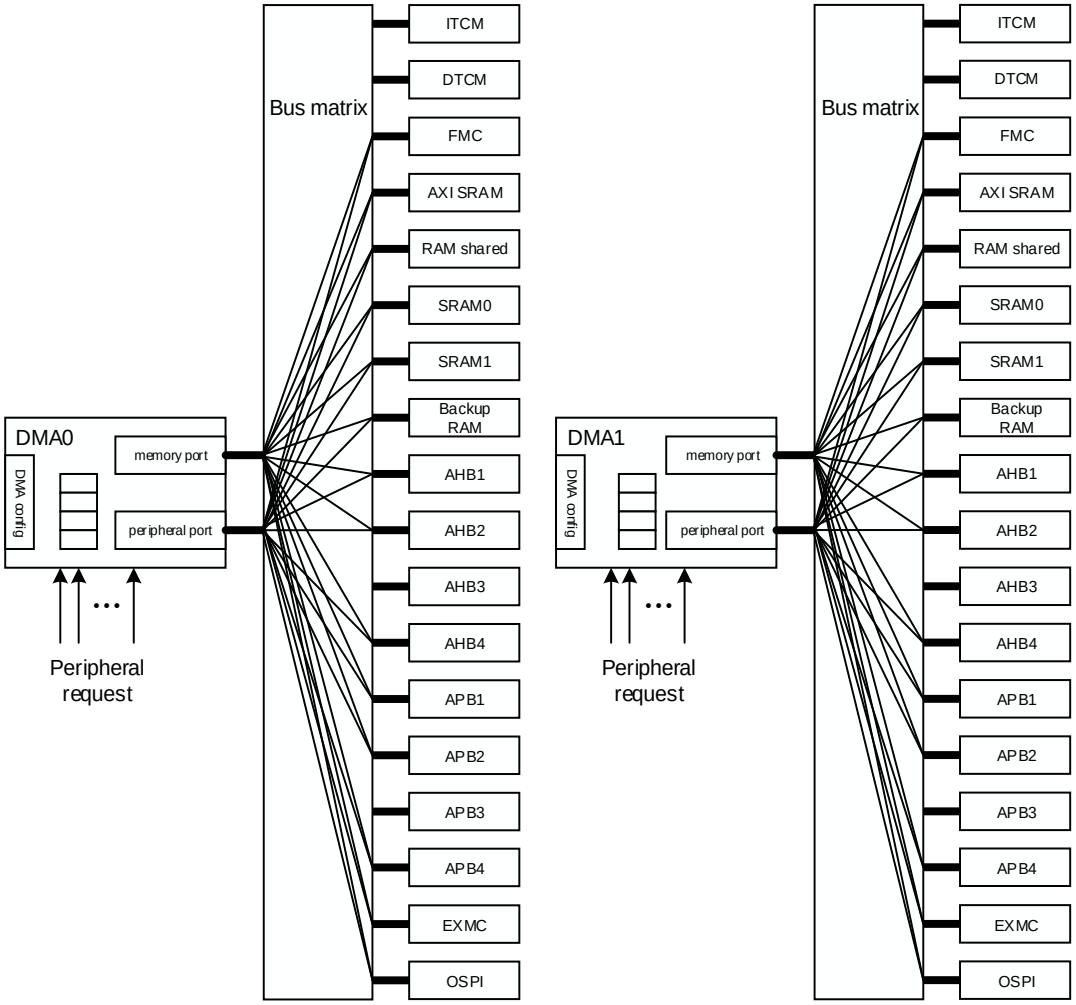
attempts to update a memory address register currently accessed by the DMA controller, a register access error is detected. For example, when the memory 0 buffer is the current source or destination, a write access on the DMA_CHxM0ADDR register could produce a register access error. When a register access error occurs, the DMA transmission is stopped when the current memory and peripheral transfer are completed and the valid FIFO data are entirely drained into the memory if needed.

When the register access error is asserted and the enabled bit for the transfer access error and exception interrupt is set, an interrupt is generated.

### Bus error

When the address accessed by the DMA controller is beyond the allowed area, a response error will be received and the channel is disabled immediately. The allowed and forbidden access region for DMA0 and DMA1 are shown in ***Figure 16-8. System connection of DMA0 and DMA1***. When the bus error is asserted and the enabled bit for the transfer access error and exception interrupt is set, an interrupt is generated.

**Figure 16-8. System connection of DMA0 and DMA1**



### 16.4.4. DMA request mapping

The DMA requests of a channel are coming from the AHB/APB peripherals through the corresponding channel output of DMAMUX request multiplexer, refers to ***Table 18-2. Request multiplexer input mapping***.

## 16.5. Register definition

DMA0 base address: 0x4002 0000

DMA1 base address: 0x4002 0400

### 16.5.1. Interrupt flag register 0 (DMA_INTF0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FTFIF3 | HTFIF3 | TAEIF3 | SDEIF3 | Reserved | FEEIF3 | FTFIF2 | HTFIF2 | TAEIF2 | SDEIF2 | Reserved | FEEIF2 |
| | | | | r | r | r | r | | r | r | r | r | r | | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | FTFIF1 | HTFIF1 | TAEIF1 | SDEIF1 | Reserved | FEEIF1 | FTFIF0 | HTFIF0 | TAEIF0 | SDEIF0 | Reserved | FEEIF0 |
| | | | | r | r | r | r | | r | r | r | r | r | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27/21/11/5 | FTFIFx | Full Transfer finish flag of channel x (x=0…3)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC0 register.<br>0: Transfer has not finished on channel x<br>1: Transfer has finished on channel x |
| 26/20/10/4 | HTFIFx | Half transfer finish flag of channel x (x=0…3)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC0 register.<br>0: Half number of transfer has not finished on channel x<br>1: Half number of transfer has finished on channel x |
| 25/19/9/3 | TAEIFx | Transfer access error flag of channel x (x=0…3)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC0 register.<br>0: Transfer access error has not occurred on channel x<br>1: Transfer access error has occurred on channel x |
| 24/18/8/2 | SDEIFx | Single data mode exception of channel x (x=0…3)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC0 register.<br>0: Single data mode exception has not occurred on channel x<br>1: Single data mode exception has occurred on channel x |
| 23/17/7/1 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 22/16/6/0 | FEEIFx | FIFO error and exception of channel x (x=0…3)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC0 register.<br>0: FIFO error or exception has not occurred on channel x<br>1: FIFO error or exception has occurred on channel x |

### 16.5.2. Interrupt flag register 1 (DMA_INTF1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FTFIF7 | HTFIF7 | TAEIF7 | SDEIF7 | Reserved | FEEIF7 | FTFIF6 | HTFIF6 | TAEIF6 | SDEIF6 | Reserved | FEEIF6 |
| | | | | r | r | r | r | | r | r | r | r | r | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FTFIF5 | HTFIF5 | TAEIF5 | SDEIF5 | Reserved | FEEIF5 | FTFIF4 | HTFIF4 | TAEIF4 | SDEIF4 | Reserved | FEEIF4 |
| | | | | r | r | r | r | | r | r | r | r | r | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27/21/11/5 | FTFIFx | Full Transfer finish flag of channel x (x=4…7)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC1 register.<br>0: Transfer has not finished on channel x<br>1: Transfer has finished on channel x |
| 26/20/10/4 | HTFIFx | Half transfer finish flag of channel x (x=4…7)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC1 register.<br>0: Half number of transfer has not finished on channel x<br>1: Half number of transfer has finished on channel x |
| 25/19/9/3 | TAEIFx | Transfer access error flag of channel x (x=4…7)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC1 register.<br>0: Transfer access error has not occurred on channel x<br>1: Transfer access error has occurred on channel x |
| 24/18/8/2 | SDEIFx | Single data mode exception of channel x (x=4…7)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC1 register.<br>0: Single data mode exception has not occurred on channel x<br>1: Single data mode exception has occurred on channel x |

| | | |
|---|---|---|
| 23/17/7/1 | Reserved | Must be kept at reset value. |
| 22/16/6/0 | FEEIFx | FIFO error and exception of channel x (x=4…7)<br>Hardware set and software cleared by writing 1 to the corresponding bit in DMA_INTC1 register.<br>0: FIFO error or exception has not occurred on channel x<br>1: FIFO error or exception has occurred on channel x |

### 16.5.3. Interrupt flag clear register 0 (DMA_INTC0)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FTFIFC3 | HTFIFC3 | TAEIFC3 | SDEIFC3 | Reserved | FEEIFC3 | FTFIFC2 | HTFIFC2 | TAEIFC2 | SDEIFC2 | Reserved | FEEIFC2 |
| | | | | w | w | w | w | | w | w | w | w | w | | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | FTFIFC1 | HTFIFC1 | TAEIFC1 | SDEIFC1 | Reserved | FEEIFC1 | FTFIFC0 | HTFIFC0 | TAEIFC0 | SDEIFC0 | Reserved | FEEIFC0 |
| | | | | w | w | w | w | | w | w | w | w | w | | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27/21/11/5 | FTFIFCx | Clear bit for Full transfer finish flag of channel x (x=0…3)<br>0: No effect<br>1: Clear full transfer finish flag |
| 26/20/10/4 | HTFIFCx | Clear bit for half transfer finish flag of channel x (x=0…3)<br>0: No effect<br>1: Clear half transfer finish flag |
| 25/19/9/3 | TAEIFCx | Clear bit for ransfer access error flag of channel x (x=0…3)<br>0: No effect<br>1: Clear transfer access error flag |
| 24/18/8/2 | SDEIFCx | Clear bit for single data mode exception of channel x (x=0…3)<br>0: No effect<br>1: Clear single data mode exception flag |
| 23/17/7/1 | Reserved | Must be kept at reset value. |
| 22/16/6/0 | FEEIFCx | Clear bit for FIFO error and exception of channel x (x=0…3)<br>0: No effect<br>1: Clear FIFO error and exception flag |

### 16.5.4. Interrupt flag clear register 1 (DMA_INTC1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FTFIFC7 | HTFIFC7 | TAEIFC7 | SDEIFC7 | Reserved | FEEIFC7 | FTFIFC6 | HTFIFC6 | TAEIFC6 | SDEIFC6 | Reserved | FEEIFC6 |
| | | | | w | w | w | w | | w | w | w | w | w | | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FTFIFC5 | HTFIFC5 | TAEIFC5 | SDEIFC5 | Reserved | FEEIFC5 | FTFIFC4 | HTFIFC4 | TAEIFC4 | SDEIFC4 | Reserved | FEEIFC4 |
| | | | | w | w | w | w | | w | w | w | w | w | | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27/21/11/5 | FTFIFCx | Clear bit for full transfer finish flag of channel x (x=4…7)<br>0: No effect<br>1: Clear full transfer finish flag |
| 26/20/10/4 | HTFIFCx | Clear bit for half transfer finish flag of channel x (x=4…7)<br>0: No effect<br>1: Clear half transfer finish flag |
| 25/19/9/3 | TAEIFCx | Clear bit for transfer access error flag of channel x (x=4…7)<br>0: No effect<br>1: Clear transfer access error flag |
| 24/18/8/2 | SDEIFCx | Clear bit for single data mode exception of channel x (x=4…7)<br>0: No effect<br>1: Clear single data mode exception flag |
| 23/17/7/1 | Reserved | Must be kept at reset value. |
| 22/16/6/0 | FEEIFCx | Clear bit for FIFO error and exception of channel x (x=4…7)<br>0: No effect<br>1: Clear FIFO error and exception flag |

### 16.5.5. Channel x control register (DMA_CHxCTL)

x = 0...7, where x is a channel number

Address offset: 0x10 + 0x18 × x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | MBURST[1:0] | | PBURST[1:0] | | Reserved | MBS | SBMEN | PRIO[1:0] | |
| | | | | | | | rw | | rw | | | rw | rw | rw | |

| 15 | 14 13 | 12 11 | 10 | 9 | 8 | 7 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PAIF | MWIDTH[1:0] | PWIDTH[1:0] | MNAGA | PNAGA | CMEN | TM[1:0] | Reserved | FTFIE | HTFIE | TAEIE | SDEIE | CHEN |
| rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24:23 | MBURST[1:0] | Transfer burst type of memory<br>Software set and clear.<br>00: single burst<br>01: INCR4 (4-beat incrementing burst)<br>10: INCR8 (8-beat incrementing burst)<br>11: INCR16 (16-beat incrementing burst)<br>These bits can not be written when CHEN is ‘1’.<br>These bits are automatically locked as ‘00’ by hardware immdiately after enable CHEN if MDMEN in the DMA_CHxFCTL register is configured to ‘0’. |
| 22:21 | PBURST[1:0] | Transfer burst type of peripheral<br>Software set and clear<br>00: single burst<br>01: INCR4 (4-beat incrementing burst)<br>10: INCR8 (8-beat incrementing burst)<br>11: INCR16 (16-beat incrementing burst)<br>These bits can not be written when CHEN is ‘1’.<br>These bits are automatically locked as ‘00’ by hardware immediately after enable CHEN if MDMEN in the DMA_CHxFCTL register is configured to ‘0’. |
| 20 | Reserved | Must be kept at reset value. |
| 19 | MBS | Memory buffer select<br>Hardware and software set, hardware and software clear.<br>0: Memory 0 is selected as memory transfer area<br>1: Memory 1 is selected as memory transfer area<br>This bit can not be written when CHEN is ‘1’.<br>During the transmission, this bit can be set and cleared by hardware at the end of transfer to indicate which memory buffer is being accessed by DMA. |
| 18 | SBMEN | Switch-buffer mode enable<br>Software set and clear.<br>0: Disable switch-buffer mode<br>1: Enable switch-buffer mode<br>This bit can not be written when CHEN is ‘1’. |
| 17:16 | PRIO[1:0] | Priority level<br>Software set and clear.<br>00: Low<br>01: Medium |

| | | |
|---|---|---|
| | | 10: High<br>11: Ultra high<br>These bits can not be written when CHEN is ‘1’. |
| 15 | PAIF | Peripheral address increment fixed<br>Software set and clear.<br>0: The peripheral address increment is determined by PWIDTH<br>1: The peripheral address increment is fixed to 4<br>This bit can not be written when CHEN is ‘1’.<br>During the transmission, when PNAGA is configured to ‘0’, this bit has no effect.<br>These bits are automatically locked as ‘0’ by hardware immediately after enable CHEN if MDMEN in the DMA_CHxFCTL register is configured to ‘0’ or PBURST are not equal to ‘00’. |
| 14:13 | MWIDTH[1:0] | Transfer width of memory<br>Software set and clear.<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: Reserved<br>These bits can not be written when CHEN is ‘1’.<br>These bits are automatically locked as PWIDTH by hardware immediately after enable CHEN if MDMEN in the DMA_CHxFCTL register is configured to ‘0’. |
| 12:11 | PWIDTH[1:0] | Transfer width of peripheral<br>Software set and clear.<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: Reserved<br>These bits can not be written when CHEN is ‘1’. |
| 10 | MNAGA | Next address generation algorithm of memory<br>Software set and clear<br>0: Fixed address mode<br>1: Increasing address mode<br>This bit can not be written when CHEN is ‘1’. |
| 9 | PNAGA | Next address generation algorithm of peripheral<br>Software set and clear<br>0: Fixed address mode<br>1: Increasing address mode<br>This bit can not be written when CHEN is ‘1’. |
| 8 | CMEN | Circular mode enable<br>Software set and clear.<br>0: Disable circular mode. |

| | | |
|---|---|---|
| | | 1: Enable circular mode<br>This bit can not be written when CHEN is '1'.<br>This bit is automatically locked as '1' by hardware immediately after enable CHEN if SBMEN is configured to '1'. |
| 7:6 | TM[1:0] | Transfer mode<br>Software set and clear.<br>00: Read from peripheral and write to memory<br>01: Read from memory and write to peripheral<br>10: Read from memory and write to memory<br>11: Reserved<br>These bits can not be written when CHEN is '1'. |
| 5 | Reserved | Must be kept at reset value. |
| 4 | FTFIE | Enable bit for full transfer finish interrupt<br>Software set and clear.<br>0:Disable full transfer finish interrupt<br>1:Enable full transfer finish interrupt |
| 3 | HTFIE | Enable bit for half transfer finish interrupt<br>Software set and clear.<br>0: Disable half transfer finish interrupt<br>1: Enable half transfer finish interrupt |
| 2 | TAEIE | Enable bit for tranfer access error interrupt<br>Software set and clear.<br>0: Disable tranfer access error interrupt<br>1: Enable tranfer access error interrupt |
| 1 | SDEIE | Enable bit for single data mode exception interrupt<br>Software set and clear.<br>0: Disable single data mode exception interrupt<br>1: Enable single data mode exception interrupt |
| 0 | CHEN | Channel enable<br>Software set, hardware clear.<br>0: Disable channel<br>1: Enable channel<br>When this bit is asserted, the DMA transfer is started. This bit is automaticly cleared when one of the following situations occurs:<br>When the transfer of channel is fully finished.<br>When a wrong FIFO configuration or a transfer access error is detected.<br>After a software clear operation, this bit is still read as 1 to indicate that there are memory or peripheral transfers still active until hardware has terminated all activity, at which point this bit is read as 0. Software can therefore poll this bit to determine when this channel is free for a new DMA transfer. |

### 16.5.6. Channel x counter register (DMA_CHxCNT)

x = 0...7, where x is a channel number

Address offset: 0x14 + 0x18 × x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CNT[15:0] | Transfer counter<br>These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’. These bits are related to PWIDTH. During the transmission, These bits signify the number of remaining data to be transferred. After each DMA peripheral transfer, CNT is decreamented by 1. If CMEN or SBMEN in the DMA_CHxCTL register is configured to ‘1’, CNT can be reloaded automatically to the original value at the end of transfer. |

### 16.5.7. Channel x peripheral base address register (DMA_CHxPADDR)

x = 0...7, where x is a channel number

Address offset: 0x18 + 0x18 × x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PADDR[31:0] | Peripheral base address<br>These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’. When PWIDTH is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address. |

When PWIDTH is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address.
Note: If PAIF in the DMA_CHxCTL register is enable, these bits must be configured to 32-bit alignment.

## 16.5.8. Channel x memory 0 base address register (DMA_CHxM0ADDR)

x = 0...7, where x is a channel number
Address offset: 0x1C + 0x18 × x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M0ADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| M0ADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | M0ADDR[31:0] | Memory 0 base address<br>When MBS in the DMA_CHxCTL register is read as to ‘0’, these bits specific the memory base address accessed by DMA during the transmission.<br>These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’ and MBS in the DMA_CHxCTL register is read as ‘0’.<br>When memory 0 is selected as memory transfer area and MWIDTH in the DMA_CHxCTL register is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address.<br>When memory 0 is selected as memory transfer area and MWIDTH in the DMA_CHxCTL register is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address. |

## 16.5.9. Channel x memory 1 base address register (DMA_CHxM1ADDR)

x = 0...7, where x is a channel number
Address offset: 0x20 + 0x18 × x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M1ADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| M1ADDR[15:0] | | | | | | | | | | | | | | | |

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | M1ADDR[31:0] | Memory 1 base address<br>When MBS in the DMA_CHxCTL register is read as to '1', these bits specific the memory base address accessed by DMA during the transmission.<br>These bits can not be written when CHEN in the DMA_CHxCTL register is '1' and MBS in the DMA_CHxCTL register is read as '1'.<br>When memorty 1 is selected as memory tranfer area and MWIDTH in the DMA_CHxCTL register is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address.<br>When memorty 1 is selected as memory tranfer area and MWIDTH in the DMA_CHxCTL register is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address. |

## 16.5.10. Channel x FIFO control register (DMA_CHxFCTL)

x = 0...7, where x is a channel number

Address offset: 0x24 + 0x18 × x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | FEEIE | Reserved | FCNT[2:0] | | | MDMEN | FCCV[1:0] | |
| | | | | | | | | rw | | r | | | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | FEEIE | Enable bit for FIFO error and exception interrupt<br>Software set and clear.<br>0: Disable FIFO error and exception interrupt<br>1: Enable FIFO error and exception interrupt |
| 6 | Reserved | Must be kept at reset value. |
| 5:3 | FCNT[2:0] | FIFO counter<br>Hardware set and clear.<br>000: No data<br>001: One word<br>010: Two words<br>011: Three words<br>100: Empty |

| | | |
|---|---|---|
| | | 101: Full<br>110~111: Reserved<br>These bits specific the number of data stored in FIFO during the transmission.<br>When MDMEN is configured to ‘0’, these bits has no meaning. |
| 2 | MDMEN | Multi-data mode enable<br>Software set and clear.<br>0: Disable Multi-data mode<br>1: Enable Multi-data mode<br>These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’.<br>These bits are automatically locked as ‘1’ by hardware immediately after enable CHEN in the DMA_CHxCTL register if TM in the DMA_CHxCTL register is configured to ‘10’. |
| 1:0 | FCCV[1:0] | FIFO counter critical value<br>Software set and clear<br>00: One word<br>01: Two Words<br>10: Three Words<br>11: Four Words<br>These bits can not be written when CHEN in the DMA_CHxCTL register is ‘1’.<br>When MDMEN is configured to ‘0’, these bits has no meaning. |

# 17. Master direct memory access controller (MDMA)

## 17.1. Overview

The master direct memory access (MDMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the MCU, thereby increasing system performance by off-loading the MCU from copying large amounts of data and avoiding frequent interrupts to serve peripherals needing more data or having available data.

An AXI master interface, an AHB master interface and two 16 depth 64-bit width FIFOs are presented in MDMA controller, which achieves a high MDMA transmission performance. The AXI interface is used for main memory and peripheral register access (system access port), and the AHB interface is used for Cortex®-M7 TCM memory access (TCM access port). MDMA can be used in combination with a DMA controller (DMA0 or DMA1). The MDMA can provide up to 16 channels. Each channel request can be selected among any request source. The built-in arbiter is used to handle priority among MDMA requests.

## 17.2. Characteristics

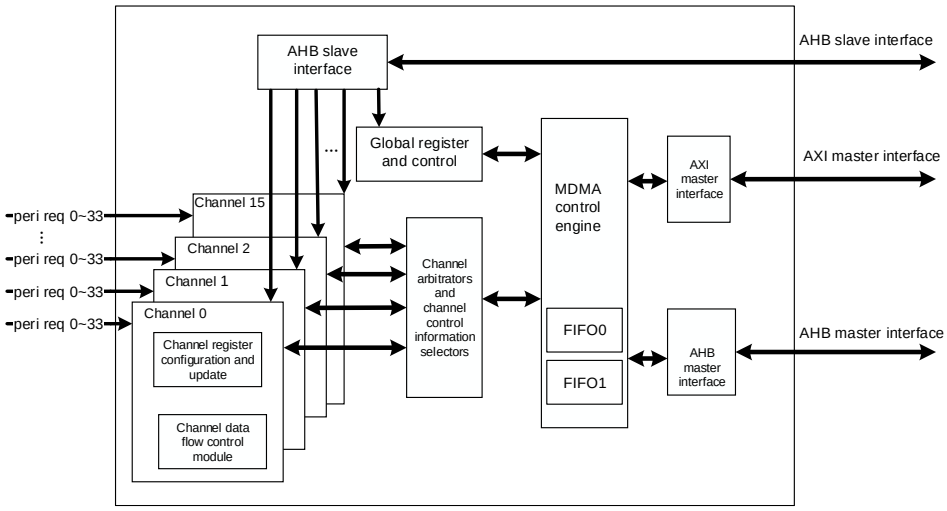
- AXI / AHB master interfaces, the AXI bus interface is used for transferring data between peripherals and memory, and the AHB bus interface is used for accessing Cortex®-M7 TCM memory.
- 16 channels and each channel supports software triggering and requests can be selected among any request source.
- Support independent single, 2, 4, 8, 16, 32, 64, 128-beat incrementing burst source and destination transfer.
- Software MDMA channel priority (low, medium, high, ultra high) and hardware MDMA channel priority (the lower the channel number, the higher the priority).
- Support independent 8, 16, 32, 64-bit source and destination transfer.
- Support independent fixed, increasing and decreasing address generation algorithm of source and destination.
- The data length and address increment of the source and destination can be configured.
- Support three transfer modes:
  - Read from memory and write to memory (software triggered).
  - Read from peripheral and write to memory (or memory mapped peripherals).
  - Read from memory (or memory mapped peripherals) and write to peripheral.
- Automatic pack / unpack of data to optimize bandwidth when the data width of the source and destination are different.
- 34 hardware trigger sources, all channels can be connected to any hardware trigger source
- Two FIFOs of 16 double word depth to maximize data bandwidth and bus utilization.
- The AHB bus interface is used to access Cortex®-M7 TCM memory. And only when the

increment and data size are identical and lower than or equal to 32-bit, burst access is allowed. When the increment and data size is larger than 32 bits, burst access is prohibited.

- Five types of event flags and independent interrupts for each channel that can be enabled and cleared.

## 17.3. Function overview

**Figure 17-1. Block diagram of MDMA**




As is shown in ***Figure 17-1. Block diagram of MDMA***, MDMA controller consists of four parts:

- AHB slave interface for MDMA configuration.
- An AXI master interface and an AHB master interface for data transmission.
- Arbiter inside to manage requests coming at the same time.
- Data processing and counting.

The MDMA controller transfers data from one address to another without CPU intervention. It supports multiple data sizes, burst types, address generation algorithm, priority levels and several transfer modes to allow for flexible application by configuring the corresponding bits in MDMA registers. All the MDMA registers can be 32-bit configured through AHB slave interface.

TRIGMOD[1:0] in the MDMA_CHxCFG register determines the data transfer mode of MDMA, as shown in ***Table 17-1. Transfer mode***.

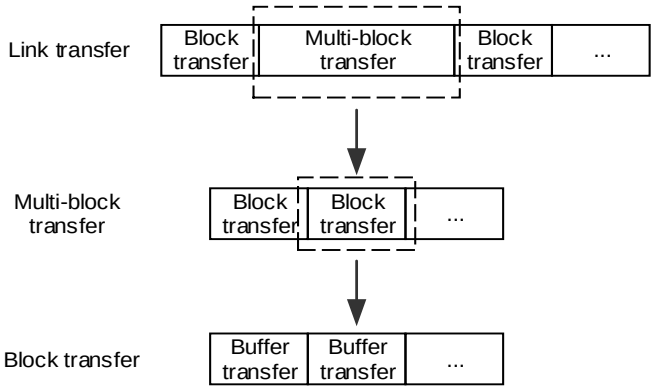
**Table 17-1. Transfer mode**

| Transfer mode | TRIGMOD[1:0] |
|---|---|
| Buffer transfer | 00 |
| Block transfer | 01 |
| Multi-block transfer | 10 |
| Link transfer | 11 |

- Buffer transfer can transmit up to 128 bytes at a time.
- Block transfer can transmit a maximum of 64KB at a time. The number of bytes to be transferred can be configured by TBNUM[16:0] in the MDMA_CHxBTCFG register. The transfer process is automatically split into multiple buffer transfers by the hardware.
- Multi-block transfer includes multiple block transfers. The number of blocks to be transferred can be configured by BRNUM[11:0] in the MDMA_CHxBTCFG register.
- Link transfer contains multiple multi-block/block transfers and the link address can be configured in the MDMA_CHxLADDR register.

The connections of the four modes is shown in ***Figure 17-2. Connections of the four modes***.

**Figure 17-2. Connections of the four modes**




The MDMA controller has 16 channels, each channel supports software triggering and can be selected between any of the request sources shown in ***Table 17-2. MDMA hardware request sources***. The channel x hardware trigger source can be selected by configuring the TRIGSEL[5:0] bit field in the MDMA_CHxCTL1 register.

**Table 17-2. MDMA hardware request sources**

| request sources TRIGSEL[5:0] | source |
|---|---|
| 0 | DMA0_CH0_TRIG |
| 1 | DMA0_CH1_TRIG |
| 2 | DMA0_CH2_TRIG |
| 3 | DMA0_CH3_TRIG |
| 4 | DMA0_CH4_TRIG |
| 5 | DMA0_CH5_TRIG |
| 6 | DMA0_CH6_TRIG |
| 7 | DMA0_CH7_TRIG |
| 8 | DMA1_CH0_TRIG |
| 9 | DMA1_CH1_TRIG |
| 10 | DMA1_CH2_TRIG |
| 11 | DMA1_CH3_TRIG |
| 12 | DMA1_CH4_TRIG |
| 13 | DMA1_CH5_TRIG |

| request sources TRIGSEL[5:0] | source |
|---|---|
| 14 | DMA1_CH6_TRIG |
| 15 | DMA1_CH7_TRIG |
| 16 | TLI_INT |
| 17 | Reserved |
| 18 | Reserved |
| 19 | Reserved |
| 20 | Reserved |
| 21 | Reserved |
| 22 | OSPI0_FT |
| 23 | OSPI0_TC |
| 24 | IPA_CLUT_TRIG |
| 25 | IPA_TC_TRIG |
| 26 | IPA_TWM_TRIG |
| 27 | Reserved |
| 28 | Reserved |
| 29 | SDIO0_DATA_END |
| 30 | SDIO0_BUF_END |
| 31 | SDIO0_CMD_END |
| 32 | OSPI1_FT |
| 33 | OSPI1_TC |

### 17.3.1. Data process

#### Arbitration

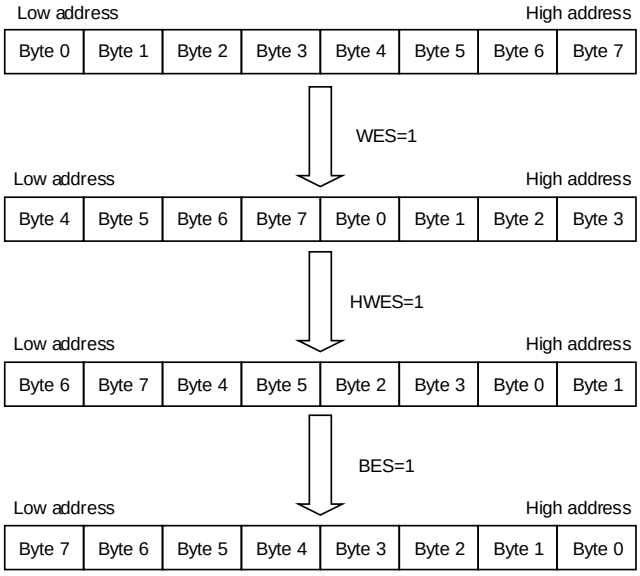
MDMA manages requests based on channel request priority through an arbiter. When more than one requests are received at the same time, the arbiter determines which channel is selected to respond according to the following priority rules:

- Software priority: Four levels, including low, medium, high and ultra high by configuring the PRIO[1:0] bits in the MDMA_CHxCTL0 register.
- Hardware priority: For channels with equal software priority level, priority is given to the channel with lower channel number. For example, if channel 0 and channel 2 are configured with the same software priority, the priority of channel 0 is higher than that of channel 2.

#### Data type

The word, halfword, and byte exchange operations on the target data can be configured by the WES/HWES/BES bits in the MDMA_CHxCTL0 register. The data exchange process is shown in ***Figure 17-3. Word, halfword, byte order exchange***.

**Figure 17-3. Word, halfword, byte order exchange**

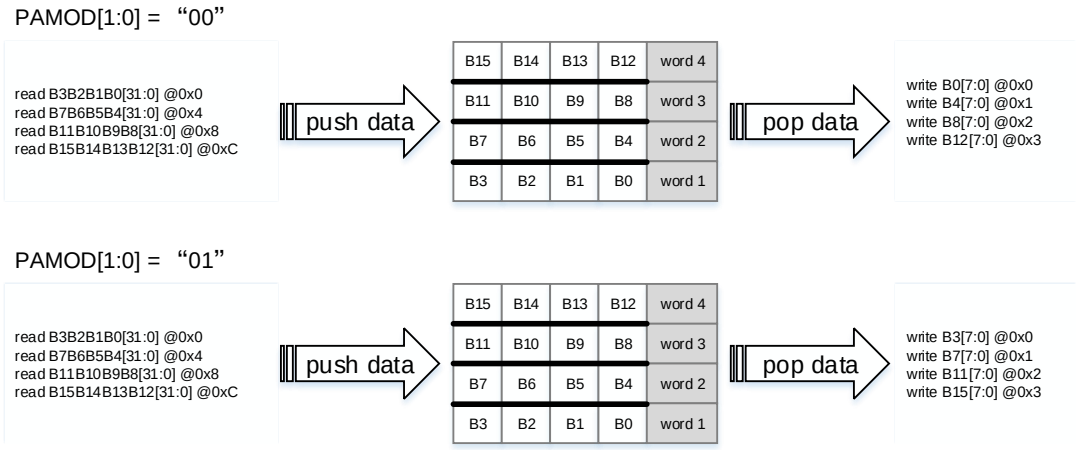



## Transfer width

The SWIDTH[1:0] and DWIDTH[1:0] bit fields of MDMA_CHxCFG register determine the data width of source and destination respectively. MDMA controller supports data widths of 8, 16, 32 and 64 bits. When PKEN is enabled and the data width SWIDTH[1:0] and DWIDTH[1:0] are not equal, MDMA packs/unpacks data automatically for data transmission in order to optimize the bandwidth. When PKEN is disabled and the data width SWIDTH[1:0] and DWIDTH[1:0] are not equal, the padding and alignment mode can be selected by configuring PAMOD[1:0] bits in the MDMA_CHxCFG register.

For example, when SWIDTH[1:0] = 10 (32 bits) and DWIDTH[1:0] = 00 (8 bits), the padding and alignment methods are shown in ***Figure 17-4. Data padding and alignment (source greater than destination)***.

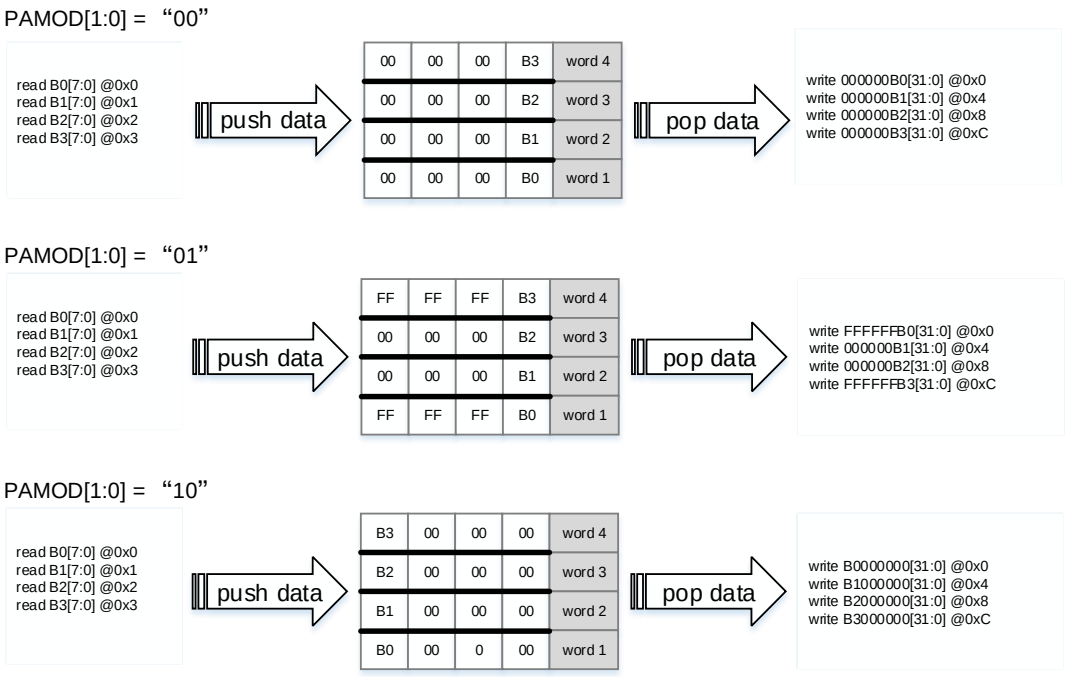
**Figure 17-4. Data padding and alignment (source greater than destination)**




Suppose the MSB of B0 and B3 is 1, and the MSB of B1 and B2 is 0, when SWIDTH[1:0] = 00 (8 bits) and DWIDTH[1:0] = 10 (32 bits), the padding and alignment methods are shown

in ***Figure 17-5. Data padding and alignment (source less than destination)***.

**Figure 17-5. Data padding and alignment (source less than destination)**

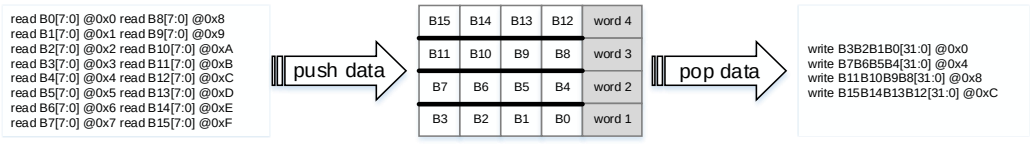



## Pack / unpack

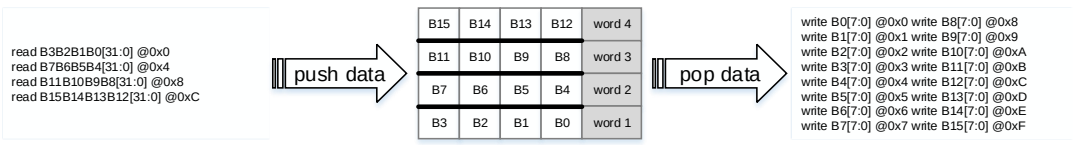
In MDMA transmission, the data size of source SWIDTH and the data size of destination DWIDTH are independent of each other, and the configuration is more flexible. When SWIDTH and DWIDTH are not equal, read/write transmission width of MDMA is different, MDMA will pack / unpack the data automatically. If the PKEN bit in MDMA_CHxCFG register is configured as 1, the source data will be packed / unpacked to match the data size of destination. When packing/unpacking data, the little endian mode is used. For example, when SWIDTH[1:0] = 00, DWIDTH[1:0] = 10, the unpacking process is shown in ***Figure 17-6. Data packing / unpacking***.

**Figure 17-6. Data packing / unpacking**

- SWIDTH[1:0] = 00, DWIDTH[1:0] = 10




- SWIDTH[1:0] = 10, DWIDTH[1:0] = 00



### Burst transmission

The SBURST[2:0] and DBURST[2:0] bits in MDMA_CHxCFG register determine the burst transmission mode of the source and destination. Both the source and destination of the MDMA supports single burst, 2-beat, 4-beat, 8-beat, 16-beat, 32-beat, 64-beat and 128-beat incrementing burst. For the single data transmission mode, SBURST[2:0] and DBURST[2:0] automatically locked as '00' by hardware immediately after the MDMA channel is enabled.

**Note:** The values of SBURST[2:0] and DBURST[2:0] must be programmed to ensure that the burst size is less than the transmission length, otherwise the results will be unpredictable.

### FIFO

The MDMA controller provides a 256 bytes buffer that is divided into two FIFOs with a depth of 16 doulbe word, and shared by all channels. The FIFOs are used to store the source data before writing it to the destination temporarily.

FIFO0 is used to store the data to be transferred of the current buffer. When the amount of data in FIFO0 meets the destination burst, MDMA will start the write operation immediately. When all the data to be transferred in the buffer has been read to FIFO0, the arbitrator begins to arbitrate the channel priority and writes the data to be transferred of the next buffer to FIFO1.

If an error occurred during the buffer transfer, the channel will be disabled and the data in FIFO0 and FIFO1 will be discarded.

## 17.3.2. Address generation

Both source and destination support three address generation algorithms: fixed mode, increasing mode, and decreasing mode independently. The DIMOD[1:0] and SIMOD[1:0] bits in MDMA_CHxCFG register are used to configure the address generation algorithm for the destination and source respectively, as is shown in ***Table 17-3. Source and destination address generation configuration***.

**Table 17-3. Source and destination address generation configuration**

| SIMOD[1:0] | | DIMOD[1:0] | |
|---|---|---|---|
| 00 | No increment | 00 | No increment |
| 10 | Increment of the source is SIOS | 10 | Increment of the destination is DIOS |
| 11 | decrement of the source is SIOS | 11 | decrement of the destination is DIOS |

In fixed mode, SIMOD[1:0] or DIMOD[1:0] is configured as "00", the address of the source and destination is always the original base address (MDMA_CHxSADDR and MDMA_CHxDADDR).

In increasing mode or decreasing mode, the source and destination address of the next transfer is the current address plus/minus 1 (or 2,4,8), depending on the configuration of

SIOS[1:0] or DIOS[1:0] in MDMA_CHxCFG.

The increment and data size can be programmed independently to optimize pack operations.

### 17.3.3. Transfer modes

#### Buffer transfer mode

MDMA supports single, 2-beat, 4-beat, 8-beat, 16-beat, 32-beat, 64-beat and 128-beat incrementing burst transmission. The burst transmission mode of the source and destination can be configured by SBURST[2:0] and DBURST[2:0] bits in MDMA_CHxCFG register. A buffer transfer is a single or burst transfer of data. SWIDTH[1:0] and DWIDTH[1:0] in MDMA_CHxCFG register are used for configuring the data width of source and destination.

When MDMA receives a request, the arbitrator manages it based on the MDMA channel request priority. If the MDMA_CHxMADDR register is not 0, the request is acknowledged when the mask data is written to the address specified by MADDR[31:0]. Or else, writing data to or reading data from the requesting peripheral will reset the request. If the request is completed by the target peripheral, BWMOD in the MDMA_CHxCFG register must be cleared to avoid wrong new MDMA requests.

If TRIGMOD[1:0] in the MDMA_CHxCFG register is 00 and the buffer transfer has completed, MDMA will wait for another request on the same channel (such as channel A).

- If another channel (such as channel B) is requested before the next request has occurred on channel A, the request from channel B will be responded regardless of whether channel B has a higher priority than channel A.
- If the next request is detected after the completion of a buffer transfer on channel A, and at the same time a request occurs on another channel (channel C), the arbitrator will manage the request event based on MDMA channel request priority.

When the buffer transfer is completed, the TCF bit in MDMA_CHxSTAT0 register will be set. The TCF bit can be cleared by writing 1 to the TCFC bit in MDMA_CHxSTATC register

If TRIGMOD[1:0] is not 00 and the total number of data to be transferred is greater than 128 bytes, then the arbitrator manages the request event based on the MDMA channel request priority after each buffer transfer. If there is no other request of higher priority, the next buffer transfer continues. If there are other requests with higher priorities, MDMA will process the higher priority requests first.

**Note:** If TRIGMOD[1:0] is 00 in buffer transfer mode:

- when BTLEN[6:0] ≥ TBNUM[15:0], number of TBNUM[15:0] bytes will be transferred.
- when BTLEN[6:0] < TBNUM[15:0], number of (BTLEN[6:0] + 1) bytes will be transferred.

#### Block transfer mode

In block transfer mode, the block size is configured by the TBNUM[16:0] in MDMA_CHxBTCFG register, and the maximum number of bytes to be transferred in the block

is 64KB. When TBNUM[16:0] counts to 0, the block transfer is completed, and TCF bit, BTCF bit and CHTCF bit in MDMA_CHxSTAT0 register will be set. The CHEN bit in the MDMA_CHxCTL0 register will be cleared by the hardware and the channel will not continue to accept MDMA requests.

In the multi-block transfer mode, if the current block is not the last block, the hardware will reload the length of the first block transfer automatically after the completion of the current block transfer, and calculate the new source address and destination address according to the values of DADDRUV and SADDRUV in MDMA_CHxMBADDRU register and SADDRUM bit and DADDRUM bit in MDMA_CHxBTCFG register, then MDMA will start the next block transfer. If the current block is the last block, when TBNUM[16:0] counts to 0 and the block transfer is complete, the TCF bit, BTCF bit, MBTCF bit and CHTCF bit in MDMA_CHxSTAT0 register will be set. The CHEN bit in the MDMA_CHxCTL0 register will be cleared by the hardware and the channel will not continue to accept MDMA requests.

In link mode, if the current block is a block transfer or the last block of multi-block transfer and MDMA_CHxLADDR is not 0, the new block configuration information is loaded according to the address specified by LADDR in the MDMA_CHxLADDR register after the current block transfer is completed, and the new block / multi-block transfer starts. If the current block is a block transfer or the last block of multi-block transfer and MDMA_CHxLADDR is 0, the TCF bit, MBTCF / BTCF bit and CHTCF bit in the MDMA_CHxSTAT0 register will be set. The CHEN bit in the MDMA_CHxCTL0 register will be cleared by the hardware and the channel will not continue to accept MDMA requests.

If the block size is not an integer multiple of the source or destination data size, the BZERR bit in MDMA_CHxSTAT1 register will be set by hardware. The BZERR bit can be cleared by writing 1 to the ERRC bit in MDMA_CHxSTATC register.

The TCF bit, BTCF bit, MBTCF bit and CHTCF bit can be cleared by writing 1 to the TCFC bit, the BTCFC bit, MBTCFC and CHTCFC bit in MDMA_CHxSTATC register respectively.

### Multi-block transfer mode

The number of blocks to be transferred can be configured by BRNUM[11:0] in the MDMA_CHxBTCFG register. When BRNUM[11:0] is not 0, the multi-block transfer mode is enabled. BRNUM[11:0] can be configured from 0 to 4095. When a block transfer is completed, the BRNUM value is reduced by 1. The source address and the destination address of the next block transfer will be update in MDMA_CHxSADDR register and MDMA_CHxDADDR register according to the update mode configured by the SADDRUM bit and DADDRUM bit in the MDMA_CHxBTCFG register. The update mode of source and destination address is shown in ***Table 17-4. Update mode of source and destination address***. TBNUM[16:0] in MDMA_CHxBTCFG register will reload the value of the first block transfer. When the last block transfer is completed, the TCF bit, BTCF bit, MBTCF bit, and CHTCF bit in the MDMA_CHxSTAT0 register will be set, the CHEN bit in MDMA_CHxCTL0 register will be will be cleared by the hardware and the channel will not continue to accept MDMA requests. The TCF bit, BTCF bit, MBTCF bit and CHTCF bit can be cleared by writing 1 to the TCFC bit, the BTCFC bit, MBTCFC and CHTCFC bit in MDMA_CHxSTATC register respectively.

**Table 17-4. Update mode of source and destination address**

| Source / destination address | Update mode configuration | Updated source and destination address |
|---|---|---|
| SADDR | SADDRUM = 0 | SADDR = SADDR + SADDRUV |
| | SADDRUM = 1 | SADDR = SADDR - SADDRUV |
| DADDR | DADDRUM = 0 | DADDR = DADDR + DADDRUV |
| | DADDRUM = 1 | DADDR = DADDR - DADDRUV |

**Note:** When the BRNUM[11:0] is 0, the last block transfer is treated as a single block transfer.

### Link transfer mode

In link mode, after multi-block/block transmission, the configuration register of the current channel includes MDMA_CHxCFG, MDMA_CHxBTCFG, MDMA_CHxSADDR, MDMA_CHxDADDR, MDMA_CHxMBADDRU, MDMA_CHxLADDR, MDMA_CHxCTL1, MDMA_CHxMADDR and MDMA_CHxMDATA will be load with the data structure at the address LADDR[31:0] defined in the MDMA_CHxLADDR register. As is shown in ***Table 17-5. Register link address***. If TRIGMOD[1:0] in MDMA_CHxCFG register is "11", the channel will accept new requests or continue transmission after the configuration register is loaded.

**Table 17-5. Register link address**

| Register | Link address |
|---|---|
| MDMA_CHxCFG | LADDR[31:0] + 0x00 |
| MDMA_CHxBTCFG | LADDR[31:0] + 0x04 |
| MDMA_CHxSADDR | LADDR[31:0] + 0x08 |
| MDMA_CHxDADDR | LADDR[31:0] + 0x0C |
| MDMA_CHxMBADDRU | LADDR[31:0] + 0x10 |
| MDMA_CHxLADDR | LADDR[31:0] + 0x14 |
| MDMA_CHxCTL1 | LADDR[31:0] + 0x18 |
| MDMA_CHxMADDR | LADDR[31:0] + 0x20 |
| MDMA_CHxMDATA | LADDR[31:0] + 0x24 |

If the TRIGSEL[5:0] in the MDMA_CHxCTL1 register changes while loading the channel configuration register, the trigger source will be changed by hardware automatically.

**Note:** In link transfer mode, the SWREQMOD bit and TRIGMOD[1:0] in the MDMA_CHxCFG register cannot be modified.

## 17.3.4. Transfer status

### Transfer complete

When TBNUM[16:0] in MDMA_CHxBTCFG register, BRNUM[11:0] and LADDR[31:0] in MDMA_CHxLADDR register are all 0, or the channel is disabled (CHEN=0) before the end of transmission, and the remaining data in FIFO are all transferred to the destination, the CHTCF bit in the MDMA_CHxSTAT0 register will be set after the channel transmission is completed.

### Transfer interrupt

Transfer interrupt means that the CHEN in the MDMA_CHxCTL0 register is disabled (CHEN=0) during transmission, and the last data transmission is not continued when the channel is enabled again. After the channel is disabled, CHTCF bit in MDMA_CHxSTAT0 register will be set when all the remaining data in FIFO are transferred to the destination. The number of bytes or blocks not transferred can be viewed in TBNUM[16:0], BRNUM[11:0] in the MDMA_CHxBTCFG register.

### Transfer pause

Before the TBNUM[16:0] in the MDMA_CHxBTCFG register counts to 0, the CHEN in MDMA_CHxCTL0 register can be cleared to pause the channel transfer. When the CHTCF bit in the MDMA_CHxSTAT0 register is set to 1, it indicates that the remaining data in the FIFO has been transmitted. If the values of MDMA_CHxBTCFG register, MDMA_CHxSADDR register and MDMA_CHxDADDR register are not modified by software, clear the CHTCF bit in MDMA_CHxSTAT0 register and enable the CHEN bit again, then the transmission continues.

**Note:** When TRIGMOD[1:0] is 11, it is recommended to configure the LADDR field in data structure of the next node as 0 to pause the channel transfer. If the channel transfer is paused by clearing the CHEN in MDMA_CHxCTL0 register, the result will not be guaranteed.

## 17.3.5. MDMA interrupts and errors

MDMA error flags are shown in ***Table 17-6. MDMA error flags***.

**Table 17-6. MDMA error flags**

| Error name | Description |
|---|---|
| BZERR | Block size error flag |
| ASERR | Address and size error flag |
| MDTERR | Mask data transmission error flag |
| LDTERR | Link data error flag |
| ERR | Transmission error flag |

The transmission error flag (ERR) will be set when the following occurs:

- A bus error occurred during MDMA read or write access.
- The location of address alignment does not match the size of the data.
- The block size is not a multiple of the data size (source and/or target).

For each MDMA channel, there are five types of interrupt events: channel transfer complete, buffer transfer complete, block transfer complete, multi-block transfer complete, and transfer error.

MDMA_CHxSTAT0 contains the flag bit for each interrupt event, register MDMA_CHxSTATC contains the flag clear bit for each interrupt event, register MDMA_CHxCTL0 contains the enable bit for each interrupt event. As is shown in ***Table 17-7. MDMA interrupt events***.

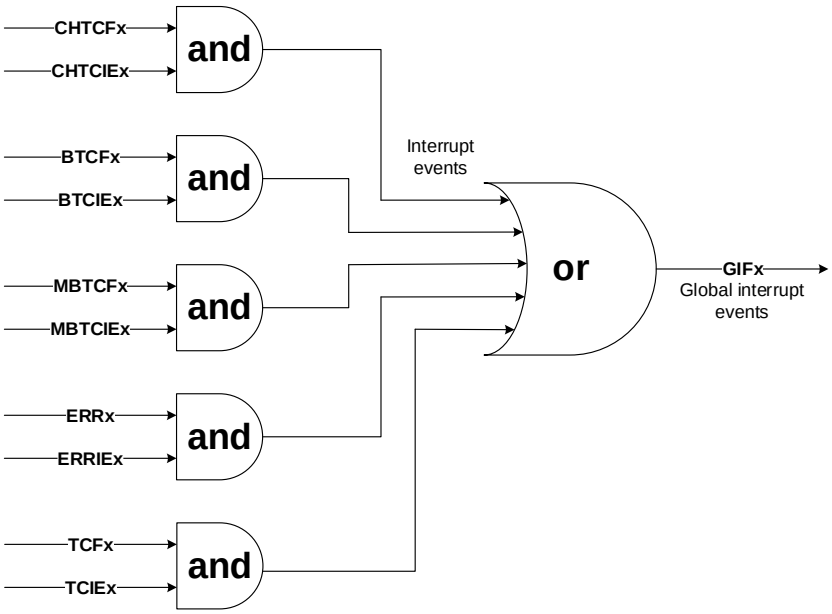
**Table 17-7. MDMA interrupt events**

| interrupt events | Flag bit | Enable bit | Clear bit |
|---|---|---|---|
| | MDMA_CHxSTAT0 | MDMA_CHxCTL0 | MDMA_CHxSTATC |
| channel transfer complete | CHTCF | CHTCIE | CHTCFC |
| buffer transfer complete | TCF | TCIE | TCFC |
| block transfer complete | BTCF | BTCIE | BTCFC |
| multi-block transfer complete | MBTCF | MBTCIE | MBTCFC |
| transfer error | ERR | ERRIE | ERRC |

If at least one of BTCF / MBTCF / CHTCF / ERR / TCF of channel x is set and the corresponding interrupt (BTCIE / MBTCIE / CHTCIE / ERRIE / TCIE) has been enabled, the GIFx in the MDMA_GINTF register will be set. If MDMA interrupt is enabled in the NVIC, an interrupt will be generated.

MDMA interrupt logic is shown in ***Figure 17-7. MDMA interrupt logic***. When any of the interrupt is enabled, and the corresponding interrupt event will generate an interrupt.

**Figure 17-7. MDMA interrupt logic**




**Note:** "x" represents the the number of channels (corresponding to x=0...15).

## 17.4. Register definition

MDMA base address: 0x52000000

### 17.4.1. Global interrupt flag register (MDMA_GINTF)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GIF15 | GIF14 | GIF13 | GIF12 | GIF11 | GIF10 | GIF9 | GIF8 | GIF7 | GIF6 | GIF5 | GIF4 | GIF3 | GIF2 | GIF1 | GIF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | GIFx | Global interrupt flag of channel x (x=0…15)<br>0: None flag of BTCF / MBTCF / CHTCF / ERR / TCF of the channel x is set, or the flag bit is set but its corresponding interrupt is disabled.<br>1: At least one of BTCF / MBTCF / CHTCF / ERR / TCF bit of channel x is set and the corresponding interrupt (BTCIE / MBTCIE / CHTCIE / ERRIE / TCIE) has been enabled. |

### 17.4.2. Channel x status register 0 (MDMA_CHxSTAT0)

x = 0...15, where x is a channel number
Address offset: 0x40 + 0x40 × x
Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | REQAF |
| | | | | | | | | | | | | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | TCF | BTCF | MBTCF | CHTCF | ERR |
| | | | | | | | | | | | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | REQAF | Channel x request active flag |

| | | |
|---|---|---|
| | | If the SWREQ bit in MDMA_CHxCTL0 is set, and CHEN is enabled, this bit will be set. When the request of channel x is completed, this bit is cleared by hardware.<br>0: Transmission of MDMA channel x is not actived.<br>1: Transmission of MDMA channel x is actived. |
| 15:5 | Reserved | Must be kept at reset value. |
| 4 | TCF | Channel x buffer transfer complete flag<br>This bit is set by hardware, and cleared by writing 1 to the corresponding bit in MDMA_CHxSTATC register.<br>0: Buffer transfer of channel x is not completed.<br>1: Buffer transfer of channel x is completed. |
| 3 | BTCF | Channel x block transfer complete flag<br>This bit is set by hardware, and cleared by writing 1 to the corresponding bit in MDMA_CHxSTATC register.<br>0: Block transfer of channel x is not completed.<br>1: Block transfer of channel x is completed. |
| 2 | MBTCF | Channel x multi-block transfer complete flag<br>This bit is set by hardware, and cleared by writing 1 to the corresponding bit in MDMA_CHxSTATC register.<br>0: Multi-block transfer of channel x is not completed.<br>1: Multi-block transfer of channel x is completed. |
| 1 | CHTCF | Channel x channel transfer complete flag<br>This bit is set by hardware, and cleared by writing 1 to the corresponding bit in MDMA_CHxSTATC register.<br>0: Channel transfer of channel x is not completed.<br>1: Channel transfer of channel x is completed. |
| 0 | ERR | Channel x transfer error flag<br>This bit is set by hardware, and cleared by writing 1 to the corresponding bit in MDMA_CHxSTATC register.<br>0: No transmission error occurred on channel x.<br>1: Transmission error occurred on channel x. |

### 17.4.3. Channel x status clear register (MDMA_CHxSTATC)

x = 0...15, where x is a channel number

Address offset: 0x44 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | TCFC | BTCFC | MBTCFC | CHTCFC | ERRC |
| | | | | | | | | | | | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | TCFC | Channel x buffer transfer complete flag clear<br>0: No effect.<br>1: Clear the TCF bit in the MDMA_CHxSTAT0 register by writing 1 to this bit. |
| 3 | BTCFC | Channel x buffer block transfer complete flag clear<br>0: No effect.<br>1: Clear the BTCF bit in the MDMA_CHxSTAT0 register by writing 1 to this bit. |
| 2 | MBTCFC | Channel x buffer multi-block transfer complete flag clear<br>0: No effect.<br>1: Clear the MBTCF bit in the MDMA_CHxSTAT0 register by writing 1 to this bit. |
| 1 | CHTCFC | Channel x channel transfer complete flag clear<br>0: No effect.<br>1: Clear the CHTCF bit in the MDMA_CHxSTAT0 register by writing 1 to this bit. |
| 0 | ERRC | Channel x transfer error flag clear<br>0: No effect.<br>1: Clear the ERR bit in the MDMA_CHxSTAT0 register by writing 1 to this bit. |

### 17.4.4. Channel x status register 1 (MDMA_CHxSTAT1)

x = 0...15, where x is a channel number

Address offset: 0x48 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | BZERR | ASERR | MDTERR | LDTERR | TERRD | ERRADDR[6:0] | | | | | | |
| | | | | r | r | r | r | r | r | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | BZERR | Block size error flag<br>When the block size or BTLEN+1 is not an integer multiple of the source or destination data size, this bit will be set by hardware. And this bit is cleared by writing |

| | | |
|---|---|---|
| | | 1 to ERRC bit in MDMA_CHxSTATC register.<br>0: No block size error is occurred.<br>1: Block size error is occurred. |
| 10 | ASERR | Address and size error flag<br>If the address does not match the data size, the bit will be set by hardware. And this bit is cleared by writing 1 to ERRC bit in MDMA_CHxSTATC register.<br>0: No address and size error is occurred.<br>1: Address and size error is occurred. |
| 9 | MDTERR | Mask data error flag<br>This bit is set by the hardware when an error occurs while writing the mask data. And this bit is cleared by writing 1 to ERRC bit in MDMA_CHxSTATC register.<br>0: No mask data error is occurred.<br>1: Mask data error is occurred. |
| 8 | LDTERR | Link data transfer error flag in the last transfer of the channel<br>The bit is set by hardware when an error occurrs while reading the link data structure. And this bit is cleared by writing 1 to ERRC bit in MDMA_CHxSTATC register.<br>0: No link data error is occurred.<br>1: Link data error is occurred. |
| 7 | TERRD | Transfer error direction<br>This bit is set by the hardware when a transmission error occurs on the channel due to write access.<br>0: Read access error.<br>1: Write access error. |
| 6:0 | ERRADDR[6:0] | Transfer error address<br>When a transfer error occurs, these bits store the low 7 bits of the error address. And the absolute error address is ERRADDR + SADDR/DADDR.<br>**Note:** These bits are ignored when link data error is occurred. |

### 17.4.5. Channel x control register 0 (MDMA_CHxCTL0)

x = 0...15, where x is a channel number

Address offset: 0x4C + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | SWREQ |
| | | | | | | | | | | | | | | | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | WES | HWES | BES | Reserved | | | SMODEN | PRIO[1:0] | | TCIE | BTCIE | MBTCIE | CHTCIE | ERRIE | CHEN |
| | rw | rw | rw | | | | rw | rw | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | SWREQ | Software request<br>When the channel is enabled, request for channel x can be activated by setting this bit. And the REQAF bit in the MDMA_CHxSTAT0 register will be set. |
| 15 | Reserved | Must be kept at reset value. |
| 14 | WES | Word endianess swapping in double word<br>0: The order of the words is not exchanged in a double word, preserving the little endian order.<br>1: Exchange the order of the words in a double word.<br>**Note:** If the destination is not a double word, the bit is ignored. When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 13 | HWES | Half word endianess swapping in word<br>0: The order of the half-words is not exchanged in a word, preserving the little endian order.<br>1: Exchange the order of the half-words in a word.<br>**Note:** If the destination is not a word or double word, the bit is ignored. When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 12 | BES | Byte endianess swapping in half word<br>0: The order of the bytes is not exchanged in a half-word, preserving the little endian order.<br>1: Exchange the order of the bytes in a half-word.<br>**Note:** If the destination is not a half-word, word or double word, the bit is ignored. When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 11:9 | Reserved | Must be kept at reset value. |
| 8 | SMODEN | Secure mode enable<br>0: Secure mode disable<br>1: Secure mode enable<br>This bit can only be written when the AHB slave port is in secure mode. If SMODEN is 0, all registers of the current channel can be written. If SMODEN is 1, all registers of the current channel are write protected.<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 7:6 | PRIO[1:0] | Priority level<br>Software set and cleared<br>00: Low<br>01: Medium<br>10: High<br>11: Ultra high<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |

| | | |
|---|---|---|
| 5 | TCIE | Buffer transfer complete interrupt enable<br>This bit is set and cleared by software.<br>0: Buffer transfer complete interrupt disable.<br>1: Buffer transfer complete interrupt enable. |
| 4 | BTCIE | Block transfer complete interrupt enable<br>This bit is set and cleared by software.<br>0: Block transfer complete interrupt disable.<br>1: Block transfer complete interrupt enable. |
| 3 | MBTCIE | Multi-block transfer complete interrupt enable<br>This bit is set and cleared by software.<br>0: Multi-block transfer complete interrupt disable.<br>1: Multi-block transfer complete interrupt enable. |
| 2 | CHTCIE | Channel transfer complete interrupt enable<br>This bit is set and cleared by software.<br>0: Channel transfer complete interrupt disable.<br>1: Channel transfer complete interrupt enable. |
| 1 | ERRIE | Transfer error interrupt enable<br>This bit is set and cleared by software.<br>0: Transfer error interrupt disable.<br>1: Transfer error interrupt enable. |
| 0 | CHEN | Channel enable<br>This bit is set and cleared by software.<br>0: channel disable<br>1: channel enable<br>**Note:** This bit will be cleared by hardware when the MDMA transfer is completed, or AHB/AXI bus error, BZERR or ASERR is occurred. |

## 17.4.6. Channel x configure register (MDMA_CHxCFG)

x = 0...15, where x is a channel number

Address offset: 0x50 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BWMOD | SWREQMOD | TRIGMOD[1:0] | | PAMOD[1:0] | | PKEN | BTLEN[6:0] | | | | | | | DBURST[2:1] | |
| rw | rw | rw | | rw | | rw | rw | | | | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DBURST[0] | SBURST[2:0] | | | DIOS[1:0] | | SIOS[1:0] | | DWIDTH[1:0] | | SWIDTH[1:0] | | DIMOD[1:0] | | SIMOD[1:0] | |
| rw | rw | | | rw | | rw | | rw | | rw | | rw | | rw | |

<table>
<tr><th>Bits</th><th>Fields</th><th>Descriptions</th></tr>
<tr><td>31</td><td>BWMOD</td><td>Bufferable write mode<br>This bit is set and cleared by software.<br>0: Bufferable write mode disable.<br>1: Bufferable write mode enable.<br>Note: When the channel is enabled (CHEN=1), this bit cannot be modified.</td></tr>
<tr><td>30</td><td>SWREQMOD</td><td>Software request mode<br>This bit is set and cleared by software.<br>0: Responds to software requests and hardware requests.<br>1: Responds to software requests.<br>Note: Changing this bit will take effect after the current transmission is completed.</td></tr>
<tr><td>29:28</td><td>TRIGMOD[1:0]</td><td>Trigger mode<br>This bit is set and cleared by software.<br>00: A software request or a hardware request triggers a buffer transfer.<br>01: A software request or a hardware request triggers a block transfer.<br>10: A software request or a hardware request triggers a multi-block transfer.<br>11: A software request or a hardware request triggers a complete data transfer (for example, link mode).<br>Note: When the channel is enabled (CHEN=1), these bits cannot be modified.</td></tr>
<tr><td>27:26</td><td>PAMOD[1:0]</td><td>Padding and alignement mode<br>This bit is set and cleared by software.
<table>
<tr><th colspan="2">size of the source data is larger than that of the destination data</th><th colspan="2">size of the source data is less than that of the destination data</th></tr>
<tr><td>00</td><td>Right aligned, write the low byte of the source to the destination address, and the high byte is discarded.</td><td>00</td><td>Right aligned, zero for the missing bits.</td></tr>
<tr><td>01</td><td>Reserved</td><td>01</td><td>Right aligned, symbol extension.</td></tr>
<tr><td>10</td><td>Reserved Left aligned, the high byte of the source is written to the target address, and the low byte is discarded.</td><td>10</td><td>Left aligned, zero for the missing bits.</td></tr>
<tr><td>11</td><td>Reserved</td><td>11</td><td>Reserved</td></tr>
</table>
Note: When the packet is enabled (PKEN=1) or the source data size is equal to the destination data size, these bits is invalid. When the channel is enabled (CHEN=1), these bits cannot be modified.</td></tr>
<tr><td>25</td><td>PKEN</td><td>Pack enable</td></tr>
</table>

| | | |
|---|---|---|
| | | This bit is set and cleared by software.<br>0: The source data is written to the destination address according to the configuration of PAMOD[1:0].<br>1: Pack/unpack the source data to match the destination data size.<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 24:18 | BTLEN[6:0] | Buffer transfer length<br>This bit is set and cleared by software.<br>The number of bytes to be transferred at a time refers to ***Buffer transfer mode***.<br>**Note:** BTLEN+1 must be a multiple of DWIDTH and SWIDTH. |
| 17:15 | DBURST[2:0] | Transfer burst type of destination<br>This bit is set and cleared by software.<br>000: single burst.<br>001: 2-beat incrementing burst.<br>010: 4-beat incrementing burst.<br>011: 8-beat incrementing burst.<br>100: 16-beat incrementing burst.<br>101: 32-beat incrementing burst.<br>110: 64-beat incrementing burst.<br>111: 128-beat incrementing burst.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 14:12 | SBURST[2:0] | Transfer burst type of source<br>These bits are set and cleared by software.<br>000: single burst.<br>001: 2-beat incrementing burst.<br>010: 4-beat incrementing burst.<br>011: 8-beat incrementing burst.<br>100: 16-beat incrementing burst.<br>101: 32-beat incrementing burst.<br>110: 64-beat incrementing burst.<br>111: 128-beat incrementing burst.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 11:10 | DIOS[1:0] | Offset size of destination increment<br>These bits are set and cleared by software.<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: 64-bit<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. If DIOS < DWIDTH and DIMOD is not 00, the result will be unpredictable. |
| 9:8 | SIOS[1:0] | Offset size of source increment<br>These bits are set and cleared by software.<br>00: 8-bit |

| | | |
|---|---|---|
| | | 01: 16-bit<br>10: 32-bit<br>11: 64-bit<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. If SIOS < SWIDTH and SIMOD is not 00, the result will be unpredictable. |
| 7:6 | DWIDTH[1:0] | Data size of destination<br>These bits are set and cleared by software.<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: 64-bit<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 5:4 | SWIDTH[1:0] | Data size of source<br>These bits are set and cleared by software.<br>00: 8-bit<br>01: 16-bit<br>10: 32-bit<br>11: 64-bit<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 3:2 | DIMOD[1:0] | Destination increment mode<br>These bits are set and cleared by software.<br>00: No increment.<br>01: Reserved.<br>10: The increment of destination address is DIOS.<br>11: The decrement of destination address is DIOS.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 1:0 | SIMOD[1:0] | Source increment mode<br>These bits are set and cleared by software.<br>00: No increment.<br>01: Reserved.<br>10: The increment of source address is SIOS.<br>11: The decrement of source address is SIOS.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |

### 17.4.7. Channel x block transfer configure register (MDMA_CHxBTCFG)

x = 0...15, where x is a channel number

Address offset: 0x54 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| BRNUM[11:0] | DADDRUM | SADDRUM | Reserved | TBNUM[16] |
|---|---|---|---|---|
| rw | rw | rw | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBNUM[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | BRNUM[11:0] | Multi-block number<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 19 | DADDRUM | Multi-block destination address update mode<br>0: DADDR = DADDR + DADDRUV<br>1: DADDR = DADDR – DADDRUV<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 18 | SADDRUM | Multi-block source address update mode<br>0: SADDR = SADDR + SADDRUV<br>1: SADDR = SADDR - SADDRUV<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 17 | Reserved | Must be kept at reset value. |
| 16:0 | TBNUM[16:0] | Transfer byte number in block<br>Number of bytes for the current block to be transferred (0-65536). In Multi-block mode, when the block transfer is completed, these bits will be reloaded with the value of the first transfer automatically.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. TBNUM must be an integer multiple of the source and target data sizes. |

### 17.4.8. Channel x source address register (MDMA_CHxSADDR)

x = 0...15, where x is a channel number

Address offset: 0x58 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SADDR[31:0] | Source address |

### 17.4.9. Channel x destination address register (MDMA_CHxDADDR)

x = 0...15, where x is a channel number

Address offset: 0x5C + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DADDR[31:0] | Destination address |

### 17.4.10. Channel x multi-block address update register (MDMA_CHxMBADDRU)

x = 0...15, where x is a channel number

Address offset: 0x60 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DADDRUV[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SADDRUV[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | DADDRUV[15:0] | Destination address update value<br>These bits are used to configure the increment or decrement of the destination address after the block transfer is completed. To align DADDR with DWIDTH, the value of these bits must be an integer multiple of DWIDTH. When BRNUM=0, these bits are invalid.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |
| 15:0 | SADDRUV[15:0] | Source address update value<br>These bits are used to configure the increment or decrement of the source address after the block transfer is completed. To align SADDR with SWIDTH, the value of these bits must be an integer multiple of SWIDTH. When BRNUM=0, these bits are invalid.<br>**Note:** When the channel is enabled (CHEN=1), these bits cannot be modified. |

### 17.4.11. Channel x link address register (MDMA_CHxLADDR)

x = 0...15, where x is a channel number

Address offset: 0x64 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | LADDR[31:0] | Link address<br>If the value of these bits is not 0, after multi-block/block transmission, the configuration register of the current channel includes MDMA_CHxCFG, MDMA_CHxBTCFG, MDMA_CHxSADDR, MDMA_CHxDADDR, MDMA_CHxMBADDRU, MDMA_CHxLADDR, MDMA_CHxCTL1, MDMA_CHxMADDR and MDMA_CHxMDATA will be load with the data structure at the address LADDR[31:0] defined in the MDMA_CHxLADDR register.<br>If the value of these bits is 0, CHTCF bit in MDMA_CHxSTAT0 register will be set, and the CHEN bit will be cleared by hardware.<br>**Note:** 1. When the channel is enabled (CHEN=1), these bits cannot be modified.<br>2. LADDR[31:0] must be double word aligned. |

### 17.4.12. Channel x control register 1 (MDMA_CHxCTL1)

x = 0...15, where x is a channel number

Address offset: 0x68 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | DBSEL | SBSEL |
| | | | | | | | | | | | | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | TRIGSEL[5:0] | | | | | |
| | | | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | DBSEL | Destination bus select |

| | | |
|---|---|---|
| | | This bit is used to configure the destination bus for the selected channel x during write operations.<br>0: The destination bus of channel x is the system bus or AXI bus.<br>1: The destination bus of channel x is AHB bus or TCM.<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 16 | SBSEL | Source bus select<br>This bit is used to configure the source bus for the selected channel x during write operations.<br>0: The source bus of channel x is the system bus or AXI bus.<br>1: The source bus of channel x is AHB bus or TCM.<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |
| 15:6 | Reserved | Must be kept at reset value. |
| 5:0 | TRIGSEL[5:0] | Trigger select<br>This bit field is used to select the hardware trigger source for channel x. If the SWREQMOD bit is 1, this bit is ignored.<br>**Note:** When the channel is enabled (CHEN=1), this bit cannot be modified. |

### 17.4.13. Channel x mask address register (MDMA_CHxMADDR)

x = 0...15, where x is a channel number

Address offset: 0x70 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MADDR[31:0] | Mask address<br>When the bit field is not 0, the DMA request is acknowledged by writing the MDATA value in the MDMA_CHxMDATA register to the address specified by MADDR. |

### 17.4.14. Channel x mask data register (MDMA_CHxMDATA)

x = 0...15, where x is a channel number

Address offset: 0x74 + 0x40 × x

Reset value: 0x0000 0000

This register has to be accessed by byte (8-bit), half-word (16-bit), word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MDATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MDATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MDATA[31:0] | Mask data |

# 18. DMA request multiplexer (DMAMUX)

## 18.1. Overview

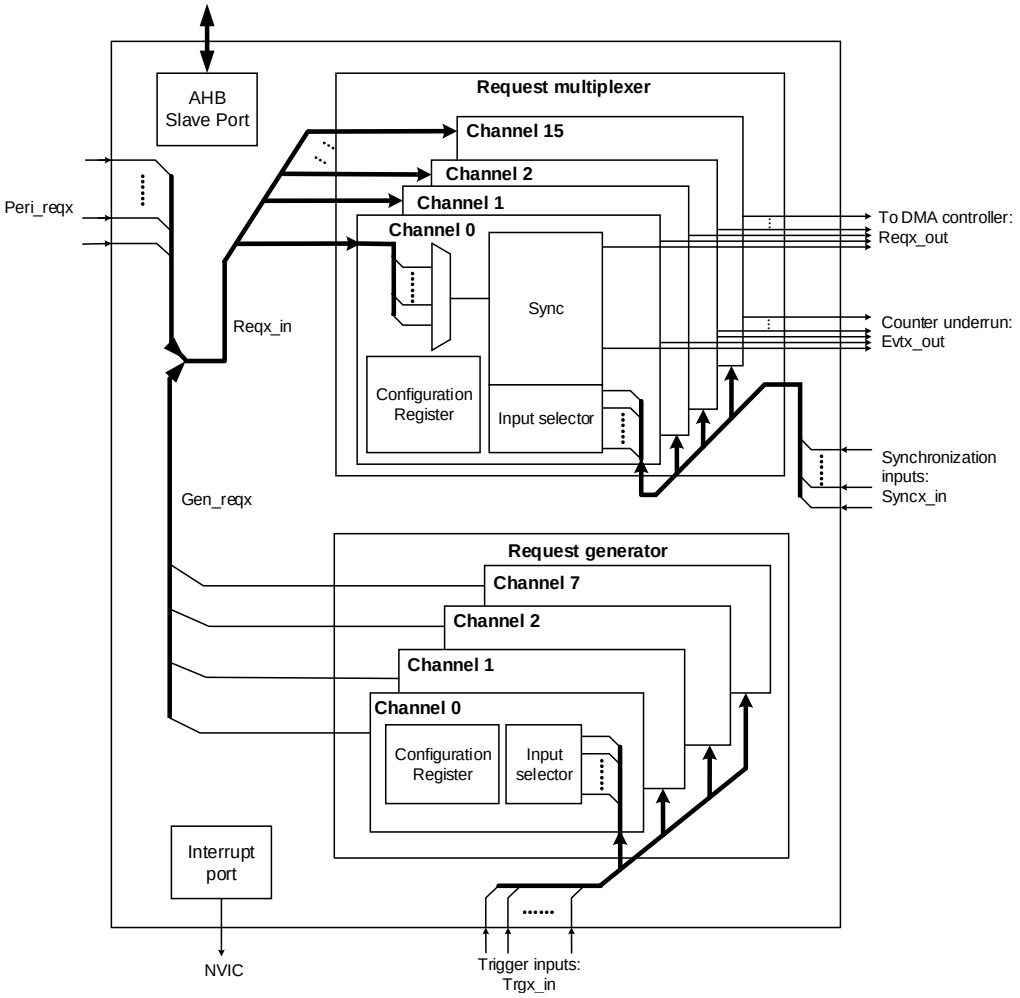
DMAMUX is a transmission scheduler for DMA requests. The DMAMUX request multiplexer is used for routing a DMA request line between the peripherals / generated DMA request (from the DMAMUX request generator) and the DMA controller. Each DMAMUX request multiplexer channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMA request is pending until it is served by the DMA controller which generates a DMA acknowledge signal (the DMA request signal is de-asserted).

## 18.2. Characteristics

- 16 channels for DMAMUX request multiplexer.
- 8 channels for DMAMUX request generator.
- Support 35 trigger inputs.
- Support 29 synchronization inputs.
- Each DMAMUX request generator channel has a DMA request trigger input selector, a DMAMUX request generator counter, and the trigger overrun flag.
- Each DMAMUX request multiplexer channel has 189 input DMA request lines from peripherals, a synchronization input selector, one DMA request line output, one channel event output for DMA request chaining, a DMAMUX request multiplexer counter, and the synchronization overrun flag.

## 18.3. Block diagram

**Figure 18-1. Block diagram of DMAMUX**




## 18.4. Signal description

The DMAMUX signals are described as follows:

- Reqx_in: DMAMUX request multiplexer inputs from peripheral requests and request generator channels.
- Peri_reqx: DMAMUX DMA request line inputs from peripherals.
- Gen_reqx: DMAMUX generated DMA request from request generator.
- Reqx_out: DMAMUX requests outputs to DMA controller.
- Trgx_in: DMAMUX DMA request triggers inputs to request generator.
- Syncx_in: DMAMUX synchronization inputs to request multiplexer.
- Evtx_out: DMAMUX request multiplexer counter underrun event outputs.

## 18.5. Function overview

As shown in ***Figure 18-1. Block diagram of DMAMUX***, DMAMUX includes two sub-blocks:

- DMAMUX request multiplexer.
  DMAMUX request multiplexer inputs (Reqx_in) source from:
  - Peripherals (Peri_reqx).
  - DMAMUX request generator outputs (Gen_reqx).

  DMAMUX request multiplexer outputs (Reqx_out) is connected to channels of DMA controller.
  Synchronization inputs (Syncx_in) source from internal or external signals.
- DMAMUX request generator.
  Trigger inputs (Trgx_in) source from internal or external signals.

### 18.5.1. DMAMUX request multiplexer

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals / generated DMA request and the DMA controllers of the product. Its component unit is the request multiplexer channels. DMA request lines are connected in parallel to all request multiplexer channels. There is a synchronization unit for each request multiplexer channel. The synchronization inputs are connected in parallel to all synchronization unit of request multiplexer channels. And there is a built-in DMAMUX request multiplexer counter for each request multiplexer channel.

#### Request multiplexer channel

A DMA request input for the DMAMUX request multiplexer channel x is configured by the MUXID[7:0] bits in the DMAMUX_RM_CHxCFG register, sourced either from the peripherals or from the DMAMUX request generator, the sources can refer to ***Table 18-2. Request multiplexer input mapping***. A DMAMUX request multiplexer channel is connected and dedicated to one single channel of the DMA controller.

**Note:** The value 0 of MUXID[7:0] bits corresponds to no DMA request line is selected. It is not allowed to configure the same DMA request line (same non-null MUXID[7:0]) to two different request multiplexer channels.

#### When synchronization mode is disabled

Each time the connected DMAMUX request is served by the DMA controller, the served DMA request is de-asserted, and the built-in DMAMUX request multiplexer counter is decremented. At the request multiplexer counter underrun, the built-in DMAMUX request multiplexer counter is automatically loaded with the value in NBR[4:0] bits of the DMAMUX_RM_CHxCFG register. If the channel event generation is enabled by setting EVGEN bit, the number of DMA requests before an output event generation is NBR[4:0] + 1.

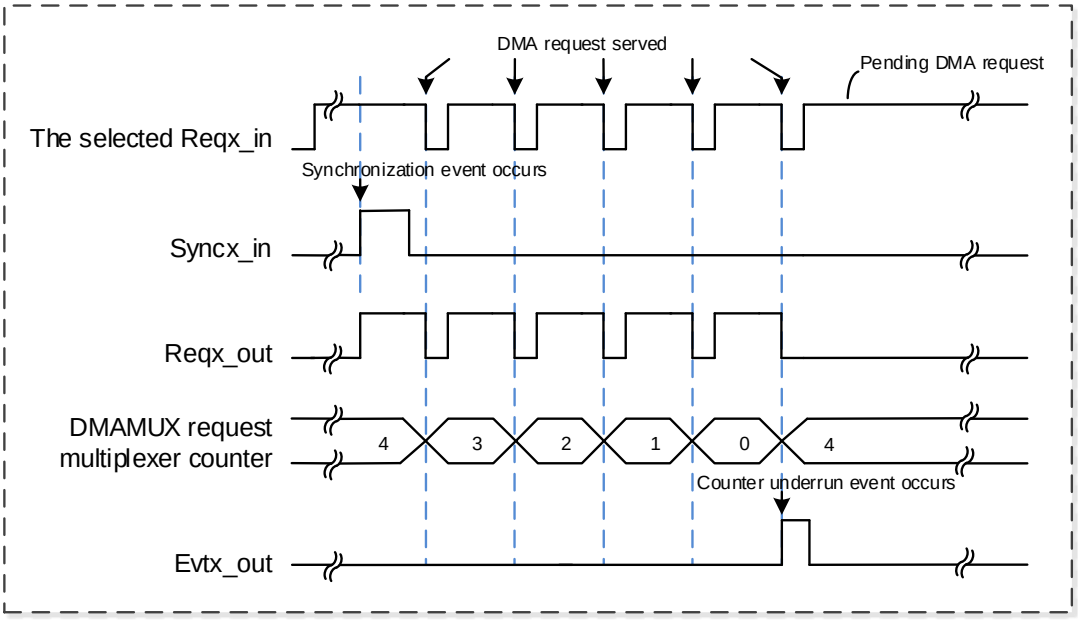
**Note:** The NBR[4:0] bits value shall only be written by software when both synchronization enable bit SYNCEN and event generation enable EVGEN bit of the corresponding request multiplexer channel x are disabled.

### When synchronization mode is enabled

A channel x in synchronization mode, when a rising/falling edge on the selected synchronization input is detected, the pending selected input DMA request line is routed to the multiplexer channel x output. Each time the connected DMAMUX request is served by the DMA controller, the served DMA request is de-asserted, and the built-in DMAMUX request multiplexer counter is decremented. At the request multiplexer counter underrun, the input DMA request line is disconnected from the request multiplexer channel x output, and the built-in DMAMUX request multiplexer counter is automatically loaded with the value in NBR[4:0] bits of the DMAMUX_RM_CHxCFG register. The number of DMA requests transferred to the request multiplexer channel x output following a detected synchronization event is NBR[4:0] + 1.

*Figure 18-2. Synchronization mode* shows an example when NBR[4:0]=4, SYNCEN=1, EVGEN=1, SYNCP[1:0]=01.

**Figure 18-2. Synchronization mode**




DMAMUX request multiplexer channel x can be synchronized by setting the synchronization enable bit SYNCEN in the DMAMUX_RM_CHxCFG register. The synchronization input is selected by SYNCID[4:0] bits in the DMAMUX_RM_CHxCFG register, the sources can refer to *Table 18-4. Synchronization input mapping*. The synchronization input valid edge is configured by the SYNCP[1:0] bits of the DMAMUX_RM_CHxCFG register.

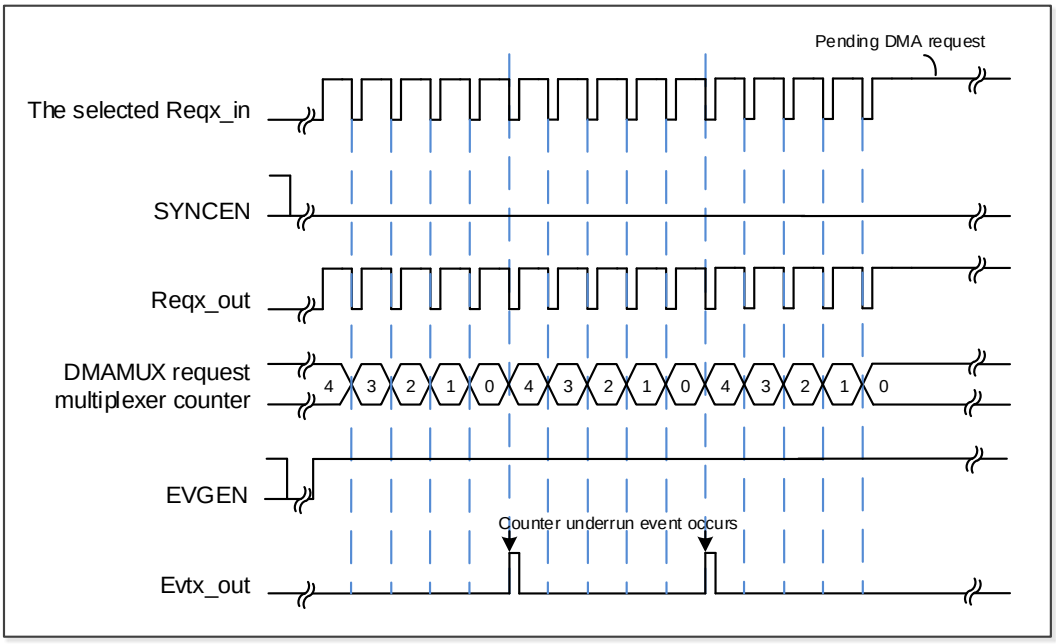
**Note:** If a synchronization input event occurs when there is no pending selected input DMA request line, the input event is discarded. The following asserted input request lines will not be routed to the DMAMUX multiplexer channel output until a synchronization input event occurs again.

### Channel event generation

Each DMA request line multiplexer channel has an event output called Evtx_out, which is the DMA request multiplexer counter underrun event. Signals Evt0_out ~ Evt3_out can be used for DMA request chaining. If event generation bit EVGEN in the DMAMUX_RM_CHxCFG register is enabled on the channel x output, when its DMA request multiplexer counter is automatically reloaded with the value of the programmed NBR[4:0] field, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle.

*Figure 18-3. Event generation* shows an example when NBR[4:0]=4, SYNCEN=0, EVGEN=1.

**Figure 18-3. Event generation**




**Note:** If EVGEN = 1 and NBR[4:0] = 0, an event is generated after each served DMA request.

### Synchronization overrun

If a new synchronization event occurs before the built-in DMAMUX request multiplexer counter underrun, the synchronization overrun flag bit SOIFx is set in the DMAMUX_RM_INTF register.

**Note:** The synchronization mode of request multiplexer channel x shall be disabled by resetting SYNCEN bit in DMAMUX_RM_CHxCFG register at the completion of the use of the related channel of the DMA controller. Otherwise, when a new synchronization event occurs, there will be a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

### 18.5.2. DMAMUX request generator

The DMAMUX request generator produces DMA requests upon trigger input event. Its component unit is the request generator channels. DMA request trigger inputs are connected in parallel to all request generator channels. And there is a built-in DMAMUX request generator counter for each request generator channel.

The active edge of trigger input events is selected through the RGTP[1:0] bits in DMAMUX_RG_CHxCFG register. The DMA request trigger input for the DMAMUX request generator channel x is selected through the TID[5:0] bits in DMAMUX_RG_CHxCFG register, the sources can refer to ***Table 18-3. Trigger input mapping***. DMAMUX request generator channel x can be enabled by setting RGEN to 1 in DMAMUX_RG_CHxCFG register.

#### Request generator channel

Upon the trigger input event, the corresponding request generator channel starts generating DMA requests on its output, and the output goes to the input of the DMAMUX request multiplexer. Each time the DMAMUX generated request is served by the connected DMA controller, the served request will be de-asserted, and the built-in DMAMUX request generator counter of the request generator channel is decremented. At the request generator counter underrun, the request generator channel stops generating DMA requests. The built-in DMAMUX request generator counter will be automatically reloaded to its programmed value upon the next trigger input event, the built-in counter is programmed by the NBRG[4:0] bits of the DMAMUX_RG_CHxCFG register.

**Note:** The number of generated DMA requests after the trigger input event is NBRG[4:0] + 1. The NBRG[4:0] value shall only be written by software when the RGEN bit of the corresponding generator channel x is disabled.

#### Trigger overrun

If a request generator channel x was enabled by RGEN bit, when a new DMA request trigger event for the request generator channel x occurs before the DMAMUX request generator counter underrun, then the request trigger overrun event flag bit TOIFx is set by hardware in the DMAMUX_RG_INTF register.

**Note:** The request generator channel x shall be disabled by resetting RGEN bit in DMAMUX_RG_CHxCFG register at the completion of the usage of the related channel of the DMA controller. Otherwise, when a new detected trigger event occurs, there will be a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

### 18.5.3. Channel configurations

The following sequence should be followed to configure a DMAMUX channel y and the related DMA channel x:

1. Set and configure the DMA channel x completely, except enabling the channel x.
2. Set and configure the related DMAMUX channel y completely.
3. Configure the CHEN bit with '1' in the DMA_CHxCTL register to enable the DMA channel x.

### 18.5.4. Interrupt

There are two types of interrupt event, including synchronization overrun event on each DMAMUX request multiplexer channel, and trigger overrun event on each DMAMUX request generator channel.

Each interrupt event has a dedicated flag bit, a dedicated clear bit, and a dedicated enable bit. The relationship is described in the following ***Table 18-1. Interrupt events***.

**Table 18-1. Interrupt events**

| Interrupt event | Flag bit | Clear bit | Enable bit |
|---|---|---|---|
| Synchronization overrun event on DMAMUX request multiplexer channel x | SOIFx in DMAMUX_RM_INTF register | SOIFCx in DMAMUX_RM_INTC register | SOIE in DMAMUX_RM_CHxCFG register |
| Trigger overrun event on DMAMUX request generator channel y | TOIFy in DMAMUX_RG_INTF register | TOIFCy in DMAMUX_RG_INTC register | TOIE in DMAMUX_RG_CHxCFG register |

#### Trigger overrun interrupt

When the DMAMUX request trigger overrun flag TOIFx is set, and the trigger overrun interrupt is enabled by setting TOIE bit, a trigger overrun interrupt will be generated. The overrun flag TOIFx is reset by writing 1 to the corresponding clear bit of overrun flag TOIFCx in the DMAMUX_RG_INTC register.

#### Synchronization overrun interrupt

When the synchronization overrun flag SOIFx is set, and the synchronization overrun interrupt is enabled by setting SOIE bit, a synchronization overrun interrupt will be generated. The overrun flag SOIFx is reset by writing 1 to the corresponding clear bit of synchronization overrun flag bit SOIFCx in the DMAMUX_RM_INTC register.

### 18.5.5. DMAMUX mapping

DMAMUX is used with DMA0 and DMA1. Channel 0 to 7 of DMAMUX are connected to channel 0 to 7 of DMA0, channel 8 to 15 of DMAMUX are connected to channel 0 to 7 of DMA1.

### Request multiplexer input mapping

A DMA request is sourced either from the peripherals or from the DMAMUX request generator, the sources can refer to ***Table 18-2. Request multiplexer input mapping***, configured by the MUXID[7:0] bits in the DMAMUX_RM_CHxCFG register for the DMAMUX request multiplexer channel x.

**Table 18-2. Request multiplexer input mapping**

| Request multiplexer channel input identification MUXID[7:0] | Source |
|---|---|
| 1 | Gen_req0 |
| 2 | Gen_req1 |
| 3 | Gen_req2 |
| 4 | Gen_req3 |
| 5 | Gen_req4 |
| 6 | Gen_req5 |
| 7 | Gen_req6 |
| 8 | Gen_req7 |
| 9 | ADC0 |
| 10 | ADC1 |
| 11 | TIMER0_CH0 |
| 12 | TIMER0_CH1 |
| 13 | TIMER0_CH2 |
| 14 | TIMER0_CH3 |
| 15 | TIMER0_MCH0 |
| 16 | TIMER0_MCH1 |
| 17 | TIMER0_MCH2 |
| 18 | TIMER0_MCH3 |
| 19 | TIMER0_UP |
| 20 | TIMER0_TRG |
| 21 | TIMER0_CMT |
| 22 | TIMER1_CH0 |
| 23 | TIMER1_CH1 |
| 24 | TIMER1_CH2 |
| 25 | TIMER1_CH3 |
| 26 | TIMER1_UP |
| 27 | TIMER1_TRG |
| 28 | Reserved |
| 29 | TIMER2_CH0 |
| 30 | TIMER2_CH1 |
| 31 | TIMER2_CH2 |
| 32 | TIMER2_CH3 |
| 33 | TIMER2_UP |

| Request multiplexer channel input identification MUXID[7:0] | Source |
|---|---|
| 34 | Reserved |
| 35 | TIMER2_TRG |
| 36 | TIMER3_CH0 |
| 37 | TIMER3_CH1 |
| 38 | TIMER3_CH2 |
| 39 | TIMER3_CH3 |
| 40 | Reserved |
| 41 | TIMER3_TRG |
| 42 | TIMER3_UP |
| 43 | I2C0_RX |
| 44 | I2C0_TX |
| 45 | I2C1_RX |
| 46 | I2C1_TX |
| 47 | SPI0_RX |
| 48 | SPI0_TX |
| 49 | SPI1_RX |
| 50 | SPI1_TX |
| 51 | USART0_RX |
| 52 | USART0_TX |
| 53 | USART1_RX |
| 54 | USART1_TX |
| 55 | USART2_RX |
| 56 | USART2_TX |
| 57 | TIMER7_CH0 |
| 58 | TIMER7_CH1 |
| 59 | TIMER7_CH2 |
| 60 | TIMER7_CH3 |
| 61 | TIMER7_MCH0 |
| 62 | TIMER7_MCH1 |
| 63 | TIMER7_MCH2 |
| 64 | TIMER7_MCH3 |
| 65 | TIMER7_UP |
| 66 | TIMER7_TRG |
| 67 | TIMER7_CMT |
| 68 | TIMER4_CH0 |
| 69 | TIMER4_CH1 |
| 70 | TIMER4_CH2 |
| 71 | TIMER4_CH3 |
| 72 | TIMER4_UP |
| 73 | Reserved |

| Request multiplexer channel input identification MUXID[7:0] | Source |
|---|---|
| 74 | TIMER4_TRG |
| 75 | SPI2_RX |
| 76 | SPI2_TX |
| 77 | UART3_RX |
| 78 | UART3_TX |
| 79 | UART4_RX |
| 80 | UART4_TX |
| 81 | DAC_CH0 |
| 82 | DAC_CH1 |
| 83 | TIMER5_UP |
| 84 | TIMER6_UP |
| 85 | USART5_RX |
| 86 | USART5_TX |
| 87 | I2C2_RX |
| 88 | I2C2_TX |
| 89 | DCI |
| 90 | CAU_IN |
| 91 | CAU_OUT |
| 92 | HAU_IN |
| 93 | UART6_RX |
| 94 | UART6_TX |
| 95 | UART7_RX |
| 96 | UART7_TX |
| 97 | SPI3_RX |
| 98 | SPI3_TX |
| 99 | SPI4_RX |
| 100 | SPI4_TX |
| 101 | SAI0_B0 |
| 102 | SAI0_B1 |
| 103 | RSPDIF_DATA |
| 104 | RSPDIF_CS |
| 105 | HPDF_FLT0 |
| 106 | HPDF_FLT1 |
| 107 | HPDF_FLT2 |
| 108 | HPDF_FLT3 |
| 109 | TIMER14_CH0 |
| 110 | TIMER14_CH1 |
| 111 | TIMER14_MCH0 |
| 112 | TIMER14_UP |
| 113 | TIMER14_TRG |

| Request multiplexer channel input identification MUXID[7:0] | Source |
|---|---|
| 114 | TIMER14_CMT |
| 115 | TIMER15_CH0 |
| 116 | TIMER15_MCH0 |
| 117 | Reserved |
| 118 | TIMER15_UP |
| 119 | TIMER16_CH0 |
| 120 | TIMER16_MCH0 |
| 121 | Reserved |
| 122 | TIMER16_UP |
| 123 | ADC2 |
| 124 | FAC_READ |
| 125 | FAC_WRITE |
| 126 | TMU_READ |
| 127 | TMU_WRITE |
| 128 | TIMER22_CH0 |
| 129 | TIMER22_CH1 |
| 130 | TIMER22_CH2 |
| 131 | TIMER22_CH3 |
| 132 | TIMER22_UP |
| 133 | Reserved |
| 134 | TIMER22_TRG |
| 135 | TIMER23_CH0 |
| 136 | TIMER23_CH1 |
| 137 | TIMER23_CH2 |
| 138 | TIMER23_CH3 |
| 139 | TIMER23_UP |
| 140 | Reserved |
| 141 | TIMER23_TRG |
| 142 | TIMER30_CH0 |
| 143 | TIMER30_CH1 |
| 144 | TIMER30_CH2 |
| 145 | TIMER30_CH3 |
| 146 | TIMER30_UP |
| 147 | Reserved |
| 148 | TIMER30_TRG |
| 149 | TIMER31_CH0 |
| 150 | TIMER31_CH1 |
| 151 | TIMER31_CH2 |
| 152 | TIMER31_CH3 |
| 153 | Reserved |

| Request multiplexer channel input identification MUXID[7:0] | Source |
|---|---|
| 154 | TIMER31_UP |
| 155 | TIMER31_TRG |
| 156 | TIMER40_CH0 |
| 157 | TIMER40_MCH0 |
| 158 | TIMER40_CMT |
| 159 | TIMER40_UP |
| 160 | TIMER41_CH0 |
| 161 | TIMER41_MCH0 |
| 162 | TIMER41_CMT |
| 163 | TIMER41_UP |
| 164 | TIMER42_CH0 |
| 165 | TIMER42_MCH0 |
| 166 | TIMER42_CMT |
| 167 | TIMER42_UP |
| 168 | TIMER43_CH0 |
| 169 | TIMER43_MCH0 |
| 170 | TIMER43_CMT |
| 171 | TIMER43_UP |
| 172 | TIMER44_CH0 |
| 173 | TIMER44_MCH0 |
| 174 | TIMER44_CMT |
| 175 | TIMER44_UP |
| 176 | TIMER50_UP |
| 177 | TIMER51_UP |
| 178 | SAI1_B0 |
| 179 | SAI1_B1 |
| 180 | SAI2_B0 |
| 181 | SAI2_B1 |
| 182 | SPI5_RX |
| 183 | SPI5_TX |
| 184 | I2C3_RX |
| 185 | I2C3_TX |
| 186 | CAN0 |
| 187 | CAN1 |
| 188 | CAN2 |
| 189 | TIMER40_CH1 |
| 190 | TIMER40_TRG |
| 191 | TIMER41_CH1 |
| 192 | TIMER41_TRG |
| 193 | TIMER42_CH1 |

| Request multiplexer channel input identification MUXID[7:0] | Source |
|---|---|
| 194 | TIMER42_TRG |
| 195 | TIMER43_CH1 |
| 196 | TIMER43_TRG |
| 197 | TIMER44_CH1 |
| 198 | TIMER44_TRG |

### Trigger input mapping

The DMA request trigger input for the DMAMUX request generator channel x is selected through the TID[5:0] bits in DMAMUX_RG_CHxCFG register, the sources can refer to ***Table 18-3. Trigger input mapping***.

**Table 18-3. Trigger input mapping**

| Trigger input identification TID[5:0] | Source |
|---|---|
| 0 | Evt0_out |
| 1 | Evt1_out |
| 2 | Evt2_out |
| 3 | Evt3_out |
| 4 | Evt4_out |
| 5 | Evt5_out |
| 6 | Evt6_out |
| 7 | EXTI_0 |
| 8 | EXTI_1 |
| 9 | EXTI_2 |
| 10 | EXTI_3 |
| 11 | EXTI_4 |
| 12 | EXTI_5 |
| 13 | EXTI_6 |
| 14 | EXTI_7 |
| 15 | EXTI_8 |
| 16 | EXTI_9 |
| 17 | EXTI_10 |
| 18 | EXTI_11 |
| 19 | EXTI_12 |
| 20 | EXTI_13 |
| 21 | EXTI_14 |
| 22 | EXTI_15 |
| 23 | RTC_WAKEUP |
| 24 | CMP0_OUTPUT |
| 25 | CMP1_OUTPUT |

| Trigger input identification TID[5:0] | Source |
|---|---|
| 26 | I2C0_WAKEUP |
| 27 | I2C1_WAKEUP |
| 28 | I2C2_WAKEUP |
| 29 | I2C3_WAKEUP |
| 30 | I2C0_INT_EVENT |
| 31 | I2C1_INT_EVENT |
| 32 | I2C2_INT_EVENT |
| 33 | I2C3_INT_EVENT |
| 34 | ADC2_INT |

**Note:** DMA requests are generated only when EXTI interrupt events occur for EXTI_x(x=0... 15).

### Synchronization input mapping

The synchronization input is selected by SYNCID[4:0] bits in the DMAMUX_RM_CHxCFG register, the sources can refer to ***Table 18-4. Synchronization input mapping***.

**Table 18-4. Synchronization input mapping**

| Synchronization input identification SYNCID[4:0] | Source |
|---|---|
| 0 | Evt0_out |
| 1 | Evt1_out |
| 2 | Evt2_out |
| 3 | Evt3_out |
| 4 | Evt4_out |
| 5 | Evt5_out |
| 6 | Evt6_out |
| 7 | EXTI_0 |
| 8 | EXTI_1 |
| 9 | EXTI_2 |
| 10 | EXTI_3 |
| 11 | EXTI_4 |
| 12 | EXTI_5 |
| 13 | EXTI_6 |
| 14 | EXTI_7 |
| 15 | EXTI_8 |
| 16 | EXTI_9 |
| 17 | EXTI_10 |
| 18 | EXTI_11 |
| 19 | EXTI_12 |
| 20 | EXTI_13 |
| 21 | EXTI_14 |

| Synchronization input identification SYNCID[4:0] | Source |
|---|---|
| 22 | EXTI_15 |
| 23 | RTC_WAKEUP |
| 24 | CMP0_OUTPUT |
| 25 | I2C0_WAKEUP |
| 26 | I2C1_WAKEUP |
| 27 | I2C2_WAKEUP |
| 28 | I2C3_WAKEUP |

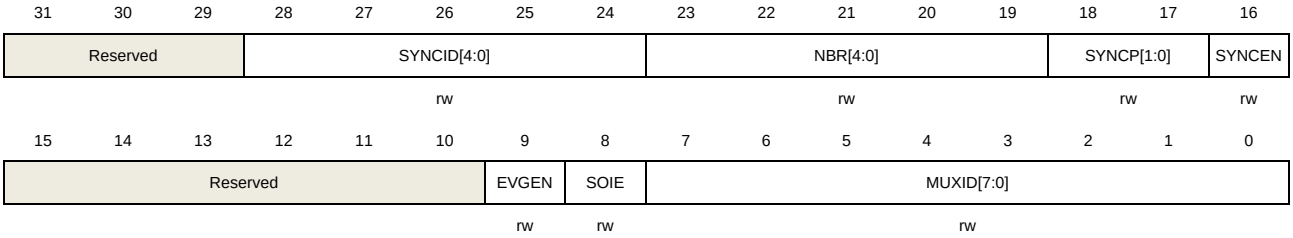
## 18.6. Register definition

DMAMUX base address: 0x4002 0800

### 18.6.1. Request multiplexer channel x configuration register (DMAMUX_RM_CHxCFG)

x = 0...15, where x is a channel number
Address offset: 0x00 + 0x04 * x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:24 | SYNCID[4:0] | Synchronization input identification<br>Selects the synchronization input source. |
| 23:19 | NBR[4:0] | Number of DMA requests to forward<br>The the number of DMA requests to forward to the DMA controller after a synchronization event / before an output event is generated equals to NBR[4:0] + 1.<br>These bits shall only be written when both SYNCEN and EVGEN bits are disabled. |
| 18:17 | SYNCP[1:0] | Synchronization input polarity<br>00: No event detection<br>01: Rising edge<br>10: Falling edge<br>11: Rising and falling edges |
| 16 | SYNCEN | Synchronization enable<br>0: Disable synchronization<br>1: Enable synchronization |
| 15:10 | Reserved | Must be kept at reset value. |
| 9 | EVGEN | Event generation enable<br>0: Disable event generation<br>1: Enable event generation |

| | | |
|---|---|---|
| 8 | SOIE | Synchronization overrun interrupt enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 7:0 | MUXID[7:0] | Multiplexer input identification<br>Selects the input DMA request in multiplexer input sources. |

### 18.6.2. Request multiplexer channel interrupt flag register (DMAMUX_RM_INTF)

Address offset: 0x80
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOIF15 | SOIF14 | SOIF13 | SOIF12 | SOIF11 | SOIF10 | SOIF9 | SOIF8 | SOIF7 | SOIF6 | SOIF5 | SOIF4 | SOIF3 | SOIF2 | SOIF1 | SOIF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | SOIFx | Synchronization overrun event flag of request multiplexer channel x<br>If a synchronization event occurs when the DMAMUX request counter value is lower than NBR[4:0], the flag is set.<br>It is cleared by writing 1 to the corresponding SOIFCx bit in DMAMUX_RM_INTC register. |

### 18.6.3. Request multiplexer channel interrupt flag clear register (DMAMUX_RM_INTC)

Address offset: 0x84
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOIFC15 | SOIFC14 | SOIFC13 | SOIFC12 | SOIFC11 | SOIFC10 | SOIFC9 | SOIFC8 | SOIFC7 | SOIFC6 | SOIFC5 | SOIFC4 | SOIFC3 | SOIFC2 | SOIFC1 | SOIFC0 |
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|

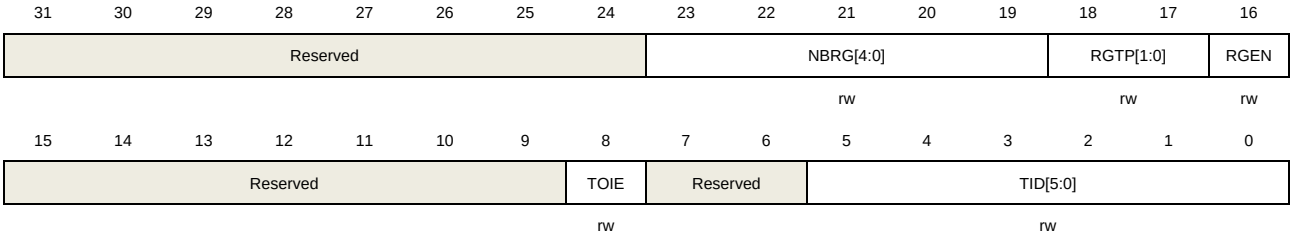
| | | |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | SOIFCx | Clear bit for synchronization overrun event flag of request multiplexer channel x. Writing 1 clears the corresponding overrun flag SOIFx in the DMAMUX_RM_INTF register. |

### 18.6.4. Request generator channel x configuration register (DMAMUX_RG_CHxCFG)

x = 0...7, where x is a channel number

Address offset: 0x100 + 0x04 * x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:19 | NBRG[4:0] | Number of DMA requests to be generated<br>The number of DMA requests to be generated after a trigger event equals to NBRG[4:0] + 1.<br>**Note:** These bits shall only be written when RGEN bit is disabled. |
| 18:17 | RGTP[1:0] | DMAMUX request generator trigger polarity<br>00: No event trigger detection<br>01: Rising edge<br>10: Falling edge<br>11: Rising and falling edges |
| 16 | RGEN | DMAMUX request generator channel x enable<br>0: Disable DMAMUX request generator channel x<br>1: Enable DMAMUX request generator channel x |
| 15:9 | Reserved | Must be kept at reset value. |
| 8 | TOIE | Trigger overrun interrupt enable<br>0: Disable interrupt<br>1: Enable interrupt |
| 7:6 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 5:0 | TID[5:0] | Trigger input identification<br>Selects the DMA request trigger input source. |

### 18.6.5. Request generator channel interrupt flag register (DMAMUX_RG_INTF)

Address offset: 0x140
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TOIF7 | TOIF6 | TOIF5 | TOIF4 | TOIF3 | TOIF2 | TOIF1 | TOIF0 |
| | | | | | | | | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | TOIFx | Trigger overrun event flag of request generator channel x.<br>If a new trigger event occurs before the request generator counter underrun, the flag is set.<br>It is cleared by writing 1 to the corresponding TOIFCx bit in the DMAMUX_RG_INTC register. |

### 18.6.6. Rquest generator channel interrupt flag clear register (DMAMUX_RG_INTC)

Address offset: 0x144
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TOIFC7 | TOIFC6 | TOIFC5 | TOIFC4 | TOIFC3 | TOIFC2 | TOIFC1 | TOIFC0 |
| | | | | | | | | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | TOIFCx | Clear bit for trigger overrun event flag of request generator channel x. |

Writing 1 clears the corresponding trigger overrun flag TOIFx in the DMAMUX_RG_INTF register.

# 19. Debug (DBG)

## 19.1. Overview

The GD32H73x_75x series provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the Arm® CoreSight™ module together with a daisy chained standard TAP controller. Debug and trace functions are integrated into the ARM® Cortex®-M7. The debug system supports serial wire debug (SWD) and trace functions in addition to standard JTAG debug. The debug and trace functions refer to the following documents:

- Cortex®-M7 Technical Reference Manual
- ARM® Debug Interface v5 Architecture Specification

The DBG hold unit helps debugger to debug in power saving mode. When corresponding bit is set, debug module provide clock when in power saving mode or hold the state for TIMER, WWDGT, FWDGT, I2C, RTC or CAN.

## 19.2. JTAG / SW function overview

Debug capabilities can be accessed by a debug tool via serial wire (SW - Debug Port) or JTAG interface (JTAG - Debug Port).

### 19.2.1. Switch JTAG or SW interface

By default, the SWD interface is active. The JTAG and SWD debug interface switching is realized through the JTAGNSW bit of the EFUSE_USER_CTL register.

### 19.2.2. Pin assignment

The JTAG interface provides 5-pin standard JTAG, known as JTAG clock (JTCK), JTAG mode selection (JTMS), JTAG data input (JTDI), JTAG data output (JTDO) and JTAG reset (NJTRST, active low). The serial wire debug (SWD) provide 2-pin SW interface, known as SW data input/output (SWDIO) and SW clock (SWCLK). The two SW pin are multiplexed with two of five JTAG pin, which is SWDIO multiplexed with JTMS, SWCLK multiplexed with JTCK. The JTDO is also used as trace async data output (TRACESWO) when async trace enabled.

**Table 19-1. Pin assignment**

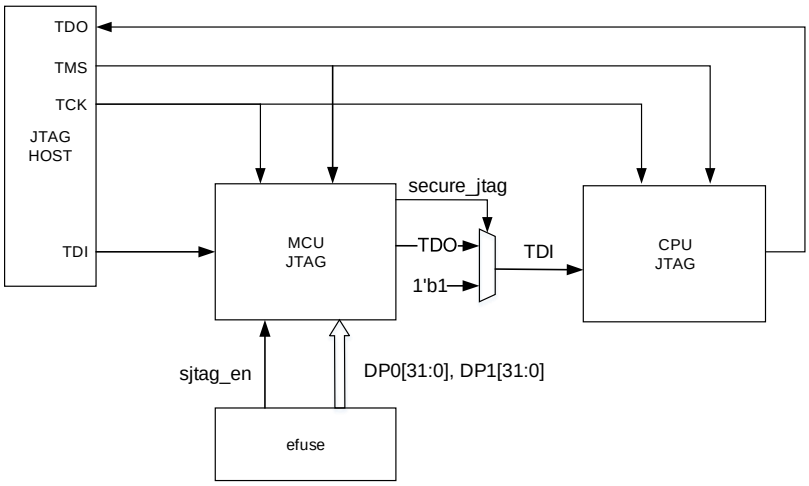
| Pin | Debug interface |
|---|---|
| PA15 | JTDI |
| PA14 | JTCK/SWCLK |
| PA13 | JTMS/SWDIO |
| PB4 | NJTRST |

| | |
|---|---|
| PB3 | JTDO |

By default, 5-pin standard JTAG debug mode is chosen after reset. Users can also use JTAG function without NJTRST pin, then the PB4 can be used to other GPIO functions (NJTRST tied to 1 by hardware). If switch to SW debug mode, the PA15/PB3/PB4 are released to other GPIO functions. If JTAG and SW not used, all 5-pin can be released to other GPIO functions.

### 19.2.3. JTAG

**Figure 19-1. Block diagram of JTAG unit**




#### JTAG daisy chained structure

The Cortex®-M7 JTAG TAP (CPU JTAG) is connected to a boundary-scan (BSD) JTAG TAP (MCU JTAG). The BSD JTAG IR is 5-bit width, while the Cortex®-M7 JTAG IR is 4-bit width. So when JTAG in IR shift step, it first shift 5-bit BYPASS instruction (5'b 11111) for BSD JTAG, and then shift normal 4-bit instruction for Cortext-M7 JTAG. Because of the data shift under BSD JTAG BYPASS mode, adding 1 extra bit to the data chain is needed.

The BSD JTAG IDCODE is 0x000717A3.

#### Secure JTAG

1. Secure JTAG only supports JTAG but not SW

2. EFUSE configuration:

EFUSE related bits: JTAGNSW, NDBG[1:0], DPx[31:0](x=0,1)

| Mode | Register configuration |
|---|---|
| No debug | NDBG[1:0] = 2b'10 or 2b'11<br>JTAGNSW: Don't care<br>DP0[31:0], DP1[31:0]: Don't care |

| | |
|---|---|
| SW | NDBG[1:0] = 2b'00 or 2b'01<br>JTAGNSW = 1b'0<br>DP0[31:0], DP1[31:0]: Don't care |
| Normal JTAG | NDBG[1:0] = 2b'00<br>JTAGNSW = 1b'1<br>DP0[31:0], DP1[31:0]: Don't care |
| Secure JTAG | NDBG[1:0] = 2b'01<br>JTAGNSW = 1b'1<br>DP0[31:0], DP1[31:0]: Efuse debug password value |

3. Using Secure JTAG

a) Configure EFUSE for secure JTAG mode: Configure DPx[31:0] as the JTAG secure password. JTAGNSW=1b'1, NDBG[1:0]= 2b'01.

b) Power reset: After the power is reset, the JTAG is in a secure state, and secure_jtag is "1". At this time, the CPU cannot be operated through JTAG.

c) Deactivate Secure JTAG: The JTAG host sequentially writes the following two passwords to the MCU JTAG to release the security mode. At this time, secure_jtag is 0, and the CPU can be operated through JTAG.

Write 5'b10101 to IR, Write DP0[31:0] to DR

Write 5'b10110 to IR, Write DP1[31:0] to DR

**Note:** 1. If the password is entered incorrectly, a power reset is required.

2. Any wrong input sequence occurrs, a power reset is required to re-decrypt.

3. Entering the correct password to open debug is limited to SPC_L and below, and will not open ROM, Flash secure mode and SPC_H.

d) Get written value and JTAG status:

Get the written value and JTAG status via JTAG:

Write 5'b11000 to IR, Read value from DR: It can be judged whether the read value is the DP0[31:0]

Write 5'b11001 to IR, Read value from DR: It can be judged whether the read value is the DP1[31:0].

Write 5'b11010 to IR, Read value from DR: {30'b0, wrong_seq, secure_jtag}, Among them, secure_jtag indicates the JTAG status. "1": The CPU cannot be operated via JTAG "0": The CPU can be operated via JTAG. wrong_seq indicates the decryption process error flag,"1": An error occurred in the decryption, "0": Decryption process without errors.

### 19.2.4. Debug reset

The JTAG-DP and SW-DP registers are in the power on reset domain. The system reset initializes the majority of the Cortex®-M7, excluding NVIC and debug logic, (FPB, DWT, and ITM). The NJTRST reset can reset JTAG TAP controller only. So, it can perform debug feature under system reset. Such as, halt-after-reset, which is the debugger sets halt under system reset, and the core halts immediately after the system reset is released.

### 19.2.5. JEDEC-106 ID code

The Cortex®-M7 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00FD000_0xE00FDFFF.

## 19.3. Debug hold function overview

### 19.3.1. Debug support for power saving mode

When STB_HOLD bit in DBG control register (DBG_CTL0) is set and entering the standby mode, the clock of AHB bus and system clock remain the same, and the debugger can debug in standby mode. When exit the standby mode, a system reset generated.

When DSLP_HOLD bit in DBG control register (DBG_CTL0) is set and entering the deep-sleep mode, the clock of AHB bus and system clock remain the same, and the debugger can debug in deep-sleep mode. When exit the deep-sleep mode, PLL is off, system clock switches to IRC64M or LPIRC4M.

When SLP_HOLD bit in DBG control register (DBG_CTL0) is set and entering the sleep mode, the clock of AHB bus for CPU is not closed, and the debugger can debug in sleep mode.

### 19.3.2. Debug support for TIMER, I2C, WWDGT, FWDGT, RTC and CAN

When the core halted and the corresponding bit in DBG control register x (DBG_CTLx, x=1,2,3,4) is set, the following behaved.

For TIMER, the timer counters stopped and hold for debug.

For I2C, SMBUS timeout hold for debug.

For WWDGT or FWDGT, the counter clock stopped for debug.

For RTC, the counter clock stopped for debug.

For CAN, the receive register stopped counting for debug.

## 19.4. Register definition

DBG base address: 0xE00E1000

### 19.4.1. ID code register (DBG_ID)

Address offset: 0x00

Read only

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID_CODE[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID_CODE[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ID_CODE[31:0] | DBG ID code register<br>These bits read by software. These bits are unchanged constant. |

### 19.4.2. Control register0 (DBG_CTL0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | TRACECLKEN | TRACE_MODE[1:0] | | Reserved | |
| | | | | | | | | | | | rw | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | STB_HOLD | DSLP_HOLD | SLP_HOLD |
| | | | | | | | | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | TRACECLKEN | Trace port clock enable<br>0: trace port clock disable<br>1: trace port clock enable. |
| 19:18 | TRACE_MODE[1:0] | Trace pin allocation mode<br>This bit is set and reset by software<br>00: Trace pin used in asynchronous mode |

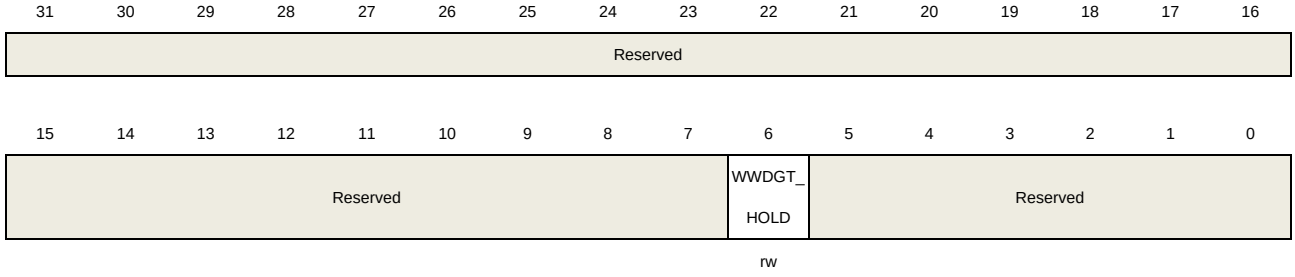
| | | |
|---|---|---|
| | | 01: Trace pin used in synchronous mode and the data length is 1<br>10: Trace pin used in synchronous mode and the data length is 2<br>11: Trace pin used in synchronous mode and the data length is 4. |
| 17:3 | Reserved | Must be kept at reset value. |
| 2 | STB_HOLD | Standby mode hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: In the standby mode, all active clocks continue to run, the debugger can debug in standby mode. |
| 1 | DSLP_HOLD | Deep-sleep mode hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1:. In the deep-sleep mode, all active clocks continue to run, the debugger can debug in deep-sleep mode. |
| 0 | SLP_HOLD | Sleep mode hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: In the seep mode, all active clocks and oscillators continue to run, the debugger can debug in deep-sleep mode. |

### 19.4.3. Control register1 (DBG_CTL1)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | WWDGT_HOLD | WWDGT hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the WWDGT counter clock for debug when core halted. |
| 5:0 | Reserved | Must be kept at reset value. |

### 19.4.4. Control register2 (DBG_CTL2)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | I2C3_HOLD | I2C2_HOLD | I2C1_HOLD | I2C0_HOLD | Reserved | | | | |
| | | | | | | | rw | rw | rw | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TIMER51_HOLD | TIMER50_HOLD | TIMER31_HOLD | TIMER30_HOLD | TIMER23_HOLD | TIMER22_HOLD | TIMER6_HOLD | TIMER5_HOLD | TIMER4_HOLD | TIMER3_HOLD | TIMER2_HOLD | TIMER1_HOLD |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | I2C3_HOLD | I2C3 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the I2C3 SMBUS timeout for debug when core halted. |
| 23 | I2C2_HOLD | I2C2 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the I2C2 SMBUS timeout for debug when core halted. |
| 22 | I2C1_HOLD | I2C1 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the I2C1 SMBUS timeout for debug when core halted. |
| 21 | I2C0_HOLD | I2C0 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the I2C0 SMBUS timeout for debug when core halted. |
| 20:12 | Reserved | Must be kept at reset value. |
| 11 | TIMER51_HOLD | TIMER51 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER51 counter for debug when core halted. |
| 10 | TIMER50_HOLD | TIMER50 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER50 counter for debug when core halted. |
| 9 | TIMER31_HOLD | TIMER31 hold bit<br>This bit is set and reset by software. |

0: no effect
1: Hold the TIMER31 counter for debug when core halted.

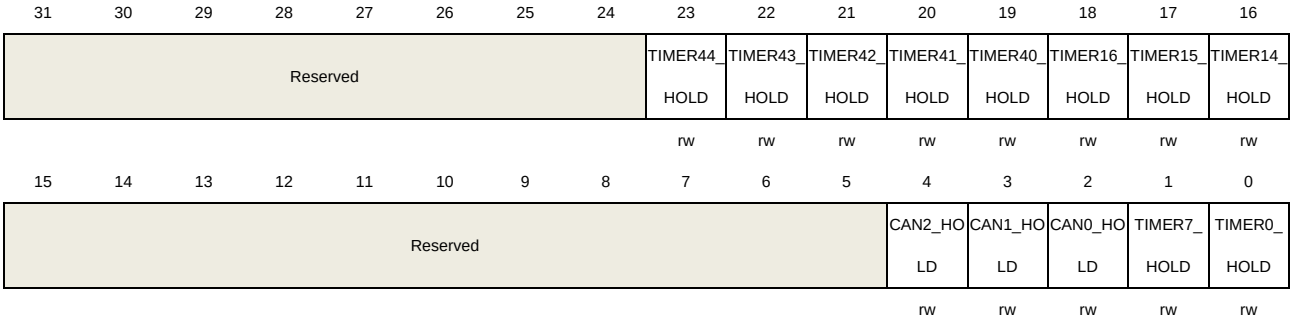
| | | |
|---|---|---|
| 8 | TIMER30_HOLD | TIMER30 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER30 counter for debug when core halted. |
| 7 | TIMER23_HOLD | TIMER23 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER23 counter for debug when core halted. |
| 6 | TIMER22_HOLD | TIMER22 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER22 counter for debug when core halted. |
| 5 | TIMER6_HOLD | TIMER6 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER6 counter for debug when core halted. |
| 4 | TIMER5_HOLD | TIMER5 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER5 counter for debug when core halted. |
| 3 | TIMER4_HOLD | TIMER4 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER4 counter for debug when core halted. |
| 2 | TIMER3_HOLD | TIMER3 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER3 counter for debug when core halted. |
| 1 | TIMER2_HOLD | TIMER2 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER2 counter for debug when core halted. |
| 0 | TIMER1_HOLD | TIMER1 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER1 counter for debug when core halted. |

### 19.4.5. Control register3 (DBG_CTL3)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | TIMER44_HOLD | TIMER44 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER44 counter for debug when core halted. |
| 22 | TIMER43_HOLD | TIMER43 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER43 counter for debug when core halted. |
| 21 | TIMER42_HOLD | TIMER42 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER42 counter for debug when core halted. |
| 20 | TIMER41_HOLD | TIMER41 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER41 counter for debug when core halted. |
| 19 | TIMER40_HOLD | TIMER40 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER40 counter for debug when core halted. |
| 18 | TIMER16_HOLD | TIMER16 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER16 counter for debug when core halted. |

| | | |
|---|---|---|
| 17 | TIMER15_HOLD | TIMER15 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER15 counter for debug when core halted. |
| 16 | TIMER14_HOLD | TIMER14 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER14 counter for debug when core halted. |
| 15:5 | Reserved | Must be kept at reset value. |
| 4 | CAN2_HOLD | CAN2 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the CAN2 for debug when core halted. |
| 3 | CAN1_HOLD | CAN1 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the CAN1 for debug when core halted. |
| 2 | CAN0_HOLD | CAN0 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the CAN0 for debug when core halted. |
| 1 | TIMER7_HOLD | TIMER7 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER7 counter for debug when core halted. |
| 0 | TIMER0_HOLD | TIMER0 hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the TIMER0 counter for debug when core halted. |

### 19.4.6. Control register4 (DBG_CTL4)

Address offset: 0x54

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | FWDGT_HOLD | Reserved | RTC_HOLD |
| | | | | | | | | | | | | | rw | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| Reserved |
|---|

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18 | FWDGT_HOLD | FWDGT hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the FWDGT counter clock for debug when core halted. |
| 17 | Reserved | Must be kept at reset value. |
| 16 | RTC_HOLD | RTC hold bit<br>This bit is set and reset by software.<br>0: no effect<br>1: Hold the RTC counter clock for debug when core halted. |
| 15:0 | Reserved | Must be kept at reset value. |

# 20. Analog-to-digital converter (ADC)

## 20.1. Overview

A 12 / 14-bit successive approximation analog-to-digital converter module(ADC) is integrated on the MCU chip. ADC0 has 20 external channels, 1 internal channel(DAC0_OUT0 channel), ADC1 has 18 external channels, 3 internal channels(the battery voltage, $V_{REFINT}$ inputs channel and DAC0_OUT1 channel), ADC2 has 17 external channels, 4 internal channels(the battery voltage, $V_{REFINT}$ inputs channel, tempeture sensor and high-precision tempeture sensor). After sampling and conversion, the conversion results can be stored in the corresponding data registers according to the least significant bit(LSB) alignment or the most significant(MSB) bit alignment(ADC0 / 1 are 32-bit data register, ADC2 is 16-bit data register). An on-chip hardware oversample scheme improves performances and reduces the computational burden of MCU.

## 20.2. Characteristics

- High performance:
  - ADC sampling rsolution:ADC0 and ADC1, 14-bit,12-bit, 10-bit or 8-bit configurable resolution. ADC2, 12-bit, 10-bit, 8-bit or 6-bit configurable resolution.
  - ADC0 and ADC1 sampling rate: 4MSPs for 14-bit resolution, 4.5 MSPs for 12-bit resolution, 5.14MSPs for 10-bit resolution, 6MSPs for 8-bit resolution, faster sampling rate can be obtained by lowering the resolution.
  - ADC2 sampling rate: 5.3 MSPs for 12-bit resolution, 6.15 MSPs for 10-bit resolution, 7.27 MSPs for 8-bit resolution, 8.89 MSPs for 6-bit resolution, faster sampling rate can be obtained by lowering the resolution.
  - Self-calibration time: ADC0 and ADC1 are 1082 ADC clock periods, ADC2 is 46 ADC clock periods.
  - Programmable sampling time.
  - Data storage mode: the most significant bit and the least significant bit
  - DMA support.
- Analog input channels:
  - 20 external analog inputs in ADC0，18 external analog inputs in ADC1，17 external analog inputs in ADC2.
  - Internal temperature sensor ($V_{SENSE}$).
  - Internal reference voltage ($V_{REFINT}$).
  - External battery power supply pin ($V_{BAT}$).
  - Internal high-precision temperature sensor ($V_{SENSE2}$).
  - Connection to DAC internal channels.
- Start-of-conversion can be initiated:
  - By software.
  - By TRIGSEL.

- Operation modes:
  - Converts a single channel or scans a sequence of channels.
  - Single operation mode converts selected inputs once per trigger.
  - Continuous operation mode converts selected inputs continuously.
  - Discontinuous operation mode.
  - SYNC mode (the device with two ADCs).
- Conversion result threshold monitor function: analog watchdog.
- Interrupt generation at the end of routine conversions, in case of analog watchdog event and overflow event.
- Oversampler:
  - 32-bit data register in ADC0 and ADC1, 16-bit data register in ADC2.
  - In ADC0 and ADC1, Oversampling ratio arbitrarily adjustable from 2x to 1024x, In ADC2, Oversampling ratio arbitrarily adjustable from 2x to 256x.
  - In ADC0 and ADC1, Programmable data shift up to 11-bit, In ADC2, Programmable data shift up to 8-bit.
- Channel input range: $V_{REFN} \leq V_{IN} \leq V_{REFP}$.
- Data can be routed to HPDF for post processing.

## 20.3. Pins and internal signals

*Figure 20-1. ADC module block diagram* shows the ADC block diagram. *Table 20-1. ADC internal input signals* and *Table 20-2. ADC input pins definition* give the ADC internal signals and pins description.

**Table 20-1. ADC internal input signals**

| Internal signal name | Description |
|---|---|
| $V_{SENSE}$ | Internal temperature sensor output voltage |
| $V_{SENSE2}$ | Internal high-precision temperature sensor2 output voltage |
| $V_{REFINT}$ | Internal voltage reference output voltage |
| $V_{BAT}$ | External battery voltage |

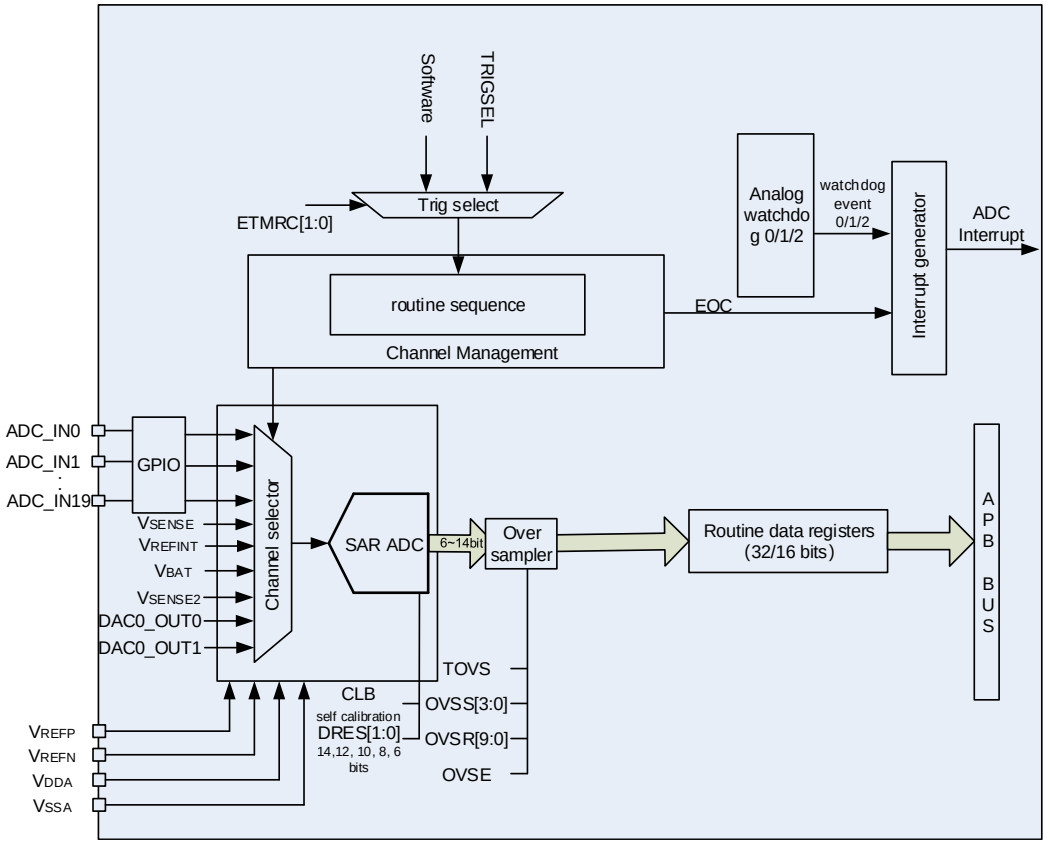
**Table 20-2. ADC input pins definition**

| Name | Description |
|---|---|
| $V_{DDA}$ | Analog power supply equal to $V_{DD}$ |
| $V_{SSA}$ | Ground for analog power supply equal to $V_{SS}$ |
| $V_{REFP}$ | The positive reference voltage for the ADC |
| $V_{REFN}$ | The negative reference voltage for the ADC |
| ADCx_IN[19:0] | Up to 20 external channels |

**Note:** $V_{DDA}$ and $V_{SSA}$ have to be connected to $V_{DD}$ and $V_{SS}$, respectively.

## 20.4. Function overview

**Figure 20-1. ADC module block diagram**




### 20.4.1. Foreground calibration function

During the foreground calibration procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is completed. Calibration should be performed before starting A/D conversion. The calibration is initiated by software by setting bit CLB=1. CLB bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon as the calibration is completed.

The calibration mode is divided into offset + mismatch and offset(only for ADC0/1), which can be modified by setting the CALMOD bit of the ADC_CTL1 register, and the offset mode is recommended.

When the ADC operating conditions change (such as supply power voltage $V_{DDA}$, positive reference voltage $V_{REFP}$, temperature and so on), it is recommended to re-run a calibration cycle.

The internal analog calibration can be reset by setting the RSTCLB bit in ADC_CTL1 register.

Calibration software procedure:

1. Ensure that ADCON=1.

2. Delay 14 CK_ADC to wait for ADC stability.
3. Set RSTCLB (optional).
4. Set CLB=1.
5. Wait until CLB=0.

### 20.4.2. Dual clock domain architecture

The CK_ADC clock provided by the clock controller is synchronous with the AHB clock. In this mode, ADCSCK[2:0] in ADC_SYNCCTL should be set different from 000.The divide factor can be 2, 4, 6, 8, 10, 12, 14,16. Thus the maximum frequency is 72 MHz for ADC0 and ADC1, 80 MHz for ADC2.

The CK_ADC can also be feed by CK_PLL1P, CK_PLL2R or CK_PER, which can be asynchronous and independent from the AHB clock. In this mode, ADCSCK[2:0] in ADC_SYNCCTL should be set to 000. The divide factor can be configured through to ADCCK[3:0] of ADC_SYNCCTL.

The RCU controller has a dedicated programmable prescaler for the ADC clock.

**Note:** The ADC1 clock shares the ADC0 clock. When using the ADC1, ADC0 clock needs to be opened, and the clock frequency division can only be configured through ADC0.

### 20.4.3. ADC enable

The ADCON bit on the ADC_CTL1 register is the enable switch of the ADC module. The ADC module will keep in reset state if this bit is 0. For power saving, when this bit is reset, the analog sub-module will be put into power off mode. After ADC is enabled, you need delay $t_{SU}$ time for sampling, the value of $t_{SU}$ please refer to the device datasheet.

### 20.4.4. Single-enden and differential input channels

By writing to bits DIFCTL[21:0] in the ADC_DIFCTL register, the user can configure channels as differential input or single-ended input, and the ADC must be disabled (ADCON = 0) when the user configurate these bits.

The channel n voltage is the difference between positive input and negative input. The positive input is external voltage $V_{INn}$, and there is a difference of the negative input between single-ended mode and differential input mode. In single-ended input mode, the negative input is $V_{REFN}$, in differential input mode, the negative input is $V_{INm}$. And therefore, channel m is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Differential channel pin shown in ***Table 20-3. ADC differential channel pin matching***.

**Table 20-3. ADC differential channel pin matching**

| Differential channel n number | ADC0 | | ADC1 | | ADC2 | |
|---|---|---|---|---|---|---|
| | $V_{INn}$ | $V_{INm}$ | $V_{INn}$ | $V_{INm}$ | $V_{INn}$ | $V_{INm}$ |
| 0 | PA0_C | PA1_C | PA0_C | PA1_C | PC2_C | PC3_C |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | PA1_C | PA0_C | PA1_C | PA0_C | PC3_C | PC2_C |
| 2 | PF11 | PF12 | PF13 | PF14 | PF9 | PF10 |
| 3 | PA6 | PA7 | PA6 | PA7 | PF7 | PF8 |
| 4 | PC4 | PC5 | PC4 | PC5 | PF5 | PF6 |
| 5 | PB1 | PB0 | PB1 | PB0 | PF3 | PF4 |
| 6 | PF12 | PF11 | PF14 | PF13 | PF10 | PF9 |
| 7 | PA7 | PA6 | PA7 | PA6 | PF8 | PF7 |
| 8 | PC5 | PC4 | PC5 | PC4 | PF6 | PF5 |
| 9 | PB0 | PB1 | PB0 | PB1 | PF4 | PF3 |
| 10 | PC0 | PC1 | PC0 | PC1 | PC0 | PC1 |
| 11 | PC1 | PC2 | PC1 | PC2 | PC1 | PC2 |
| 12 | PC2 | PC3 | PC2 | PC3 | PC2 | PC1 |
| 13 | PC3 | PC2 | PC3 | PC2 | PH2 | PH3 |
| 14 | PA2 | PA3 | PA2 | PA3 | PH3 | PH4 |
| 15 | PA3 | PA2 | PA3 | PA2 | PH4 | PH5 |
| 16 | PA0 | PA1 | null | null | PH5 | PH4 |
| 17 | PA1 | PA0 | null | null | null | null |
| 18 | PA4 | PA5 | PA4 | PA5 | null | null |
| 19 | PA5 | PA4 | PA5 | PA4 | null | null |
| 20 | null | null | null | null | null | null |
| 21 | null | null | null | null | null | null |

When the channel is used in differential input mode, the input voltages should be differential signals (common mode voltage is $V_{REFP}$/ 2), and the input ranges are still ($V_{REFN}$ ~ $V_{REFP}$).

Taking the right-aligned, 12-bit resolution as an example:

1) When $V_{INn}$ is $V_{REFP}$ and $V_{INm}$ is $V_{REFN}$, the conversion result of channel n is 0x0FFF;
2) When $V_{INn}$ is $V_{REFN}$ and $V_{INm}$ is $V_{REFP}$, The conversion result of channel n is 0x0000;
3) When $V_{INn}$ is $V_{REFP}$ / 2 and $V_{INm}$ is $V_{REFP}$ / 2, the conversion result of channel n is 0x07FF.

$D_{out}$ is the conversion result of channel n, then the differential voltage is:

$$V_{INn}-V_{INm} = V_{REFP}*(2*D_{out}/4095-1) \quad (20\text{-}1)$$

### 20.4.5. Routine sequence

The channel management circuit can organize the sampling conversion channels into a sequence: routine sequence. The routine sequence supports up to 21 channels, and each channel is called routine channel.

The ADC_RSQ0~ADC_RSQ8 registers specify the selected channels of the routine sequence. The RL[3:0] bits in the ADC_RSQ0 register specify the total conversion sequence length.

**Note:** Although the ADC supports 21 multiplexed channels, the maximum length of the

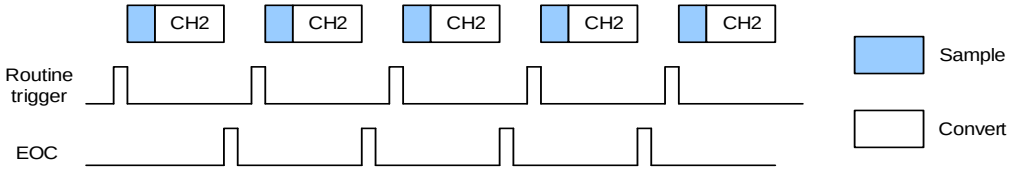
sequence is only 16.

## 20.4.6. Operation modes

### Single operation mode

In the single operation mode, the ADC performs conversion on the channel specified in the RSQ0[4:0] bits of ADC_RSQ8 register at a routine trigger. When the ADCON has been set high, the ADC samples and converts a single channel, once the corresponding software trigger or TRIGSEL trigger is active.

**Figure 20-2. Single operation mode**




After conversion of a single routine channel, the conversion data will be stored in the ADC_RDATA register, the EOC will be set. An interrupt will be generated if the EOCIE bit is set.

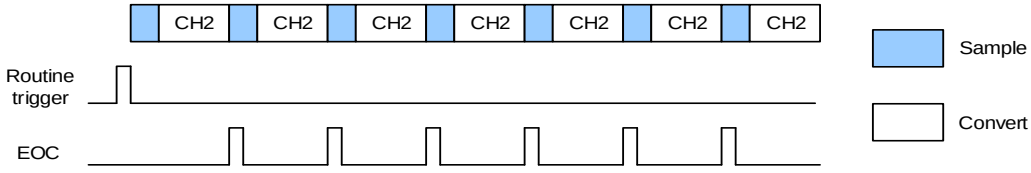
Software procedure for a single operation mode of a routine channel:

1. Make sure the DISRC, SM in the ADC_CTL0 register and CTN bit in the ADC_CTL1 register are reset.
2. Configure RSQ0 with the analog channel number.
3. Configure ADC_RSQx register.
4. Configure ETMRC[1:0] bits in the ADC_CTL1 register if in need.
5. Set the SWRCST bit, or generate an TRIGSEL trigger for the routine sequence.
6. Wait the EOC flag to be set.
7. Read the converted data in the ADC_RDATA register.
8. Clear the EOC flag by writing 0 to it.

### Continuous operation mode

The continuous operation mode will be enabled when CTN bit in the ADC_CTL1 register is set. In this mode, the ADC performs conversion on the channel specified in the RSQ0. When the ADCON has been set high, the ADC samples and converts specified channel, once the corresponding software trigger or TRIGSEL trigger is active. The conversion data will be stored in the ADC_RDATA register.

**Figure 20-3. Continuous operation mode**




Software procedure for continuous operation mode on a routine channel:

1. Set the CTN bit in the ADC_CTL1 register.
2. Configure RSQ0 with the analog channel number.
3. Configure ADC_RSQx register.
4. Configure ETMRC[1:0] bits in the ADC_CTL1 register if in need.
5. Set the SWRCST bit, or generate an TRIGSEL trigger for the routine sequence.
6. Wait the EOC flag to be set.
7. Read the converted data in the ADC_RDATA register.
8. Clear the EOC flag by writing 0 to it.
9. Repeat steps 6~8 as soon as the conversion is in need.

To get rid of checking, DMA can be used to transfer the converted data:

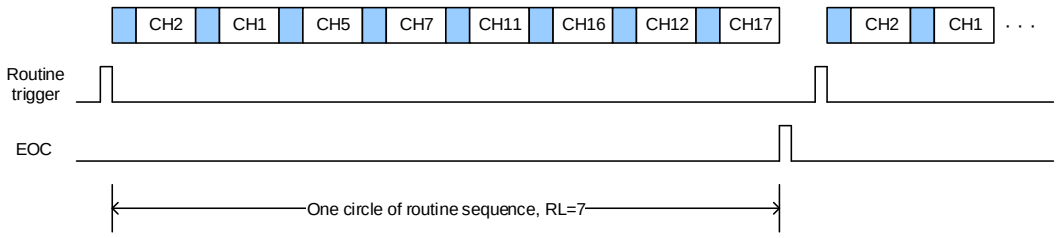
1. Set the CTN and DMA bit in the ADC_CTL1 register.
2. Configure RSQ0 with the analog channel number.
3. Configure ADC_RSQx register.
4. Configure ETMRC[1:0] bits in the ADC_CTL1 register if in need.
5. Prepare the DMA module to transfer data from the ADC_RDATA (refer to the spec of the DMA module).
6. Set the SWRCST bit, or generate an TRIGSEL trigger for the routine sequence.

### Scan operation mode

The scan operation mode will be enabled when SM bit in the ADC_CTL0 register is set. In this mode, the ADC performs conversion on all channels with a specific routine sequence specified in the ADC_RSQ0~ADC_RSQ8 registers. When the ADCON has been set high, the ADC samples and converts specified channels one by one in the routine sequence till the end of the sequence, once the corresponding software trigger or TRIGSEL trigger is active. The conversion data will be stored in the ADC_RDATA register. After conversion of the routine sequence, the EOC will be set. An interrupt will be generated if the EOCIE bit is set. The DMA bit in ADC_CTL1 register must be set when the routine sequence works in scan mode.

After conversion of a routine sequence, the conversion can be restarted automatically if the CTN bit in the ADC_CTL1 register is set.

**Figure 20-4. Scan operation mode, continuous disable**

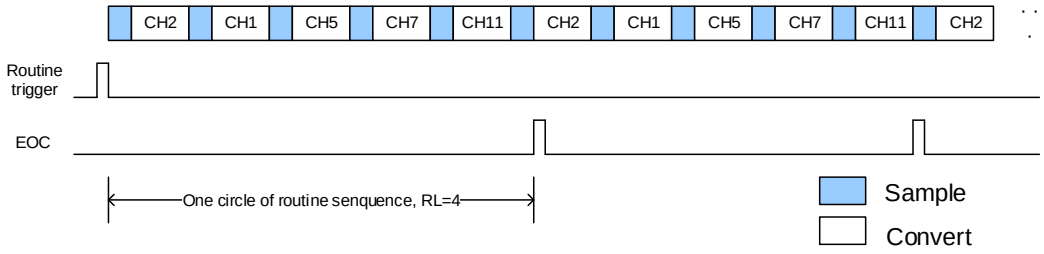



Software procedure for scan operation mode on a routine sequence:

1. Set the SM bit in the ADC_CTL0 register and the DMA bit in the ADC_CTL1 register
2. Configure ADC_RSQx registers.
3. Configure ETMRC[1:0] bits in the ADC_CTL1 register if in need.
4. Prepare the DMA module to transfer data from the ADC_RDATA (refer to the spec of the DMA module).

5. Set the SWRCST bit, or generate an TRIGSEL trigger for the routine sequence.

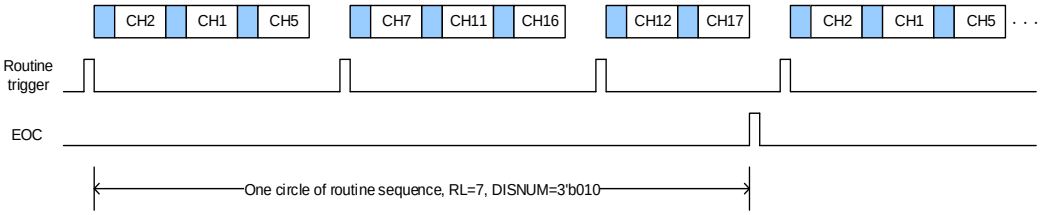
**Figure 20-5. Scan operation mode, continuous enable**




### Discontinuous operation mode

The discontinuous operation mode will be enabled when DISRC bit in the ADC_CTL0 register is set. In this mode, the ADC performs a short sequence of n conversions (n does not exceed 8) which is a part of the sequence selected in the ADC_RSQ0~ADC_RSQ8 registers. The value of n is configured by the DISNUM[2:0] bits in the ADC_CTL0 register. When the corresponding software trigger or TRIGSEL trigger is active, the ADC samples and converts the next n channels configured in the ADC_RSQ0~ADC_RSQ8 registers until all the channels of routine sequence are done. The EOC will be set after every circle of the routine sequence. An interrupt will be generated if the EOCIE bit is set.

**Figure 20-6. Discontinuous operation mode**




Software procedure for discontinuous operation mode on a routine sequence:

1. Set the DISRC bit in the ADC_CTL0 register and the DMA bit in the ADC_CTL1 register.
2. Configure DISNUM[2:0] bits in the ADC_CTL0 register.
3. Configure ADC_RSQx registers.
4. Configure ETMRC[1:0] bits in the ADC_CTL1 register if in need.
5. Prepare the DMA module to transfer data from the ADC_RDATA (refer to the spec of the DMA module).
6. Set the SWRCST bit, or generate an TRIGSEL trigger for the routine sequence.
7. Repeat step6 if in need.

## 20.4.7. Conversion result threshold monitor function

### Analog watchdog 0

The analog watchdog 0 is enabled when the RWD0EN bit in the ADC_CTL0 register is set for routine sequence. This function is used to monitor whether the conversion result exceeds the set thresholds, and when the analog voltage converted by the ADC is below a low

threshold or above a high threshold, the WDE0 bit in ADC_STAT register will be set. An interrupt will be generated if the WDE0IE bit is set. The ADC_WDHT0 and ADC_WDLT0 registers are used to specify the high and low threshold. The comparison is done before the alignment, so the threshold values are independent of the alignment, which is specified by the DAL bit in the ADC_CTL1 register. One or more channels, which are select by the RWD0EN, WD0SC and WD0CHSEL[4:0] bits in ADC_CTL0 register, can be monitored by the analog watchdog 0.

### Analog watchdog 1/2

The analog watchdog 1/2 are more flexible, and can configure the watchdog function of single or serveral channels.

The analog watchdog 1 function can be enabled by configuring the corresponding bits in the AWD1CS [21: 0] bits in the ADC_WD1SR register. Similarly, the watchdog 2 function can be configured. The high / low threshold of the analog watchdog 1/2 can be configured in the ADC_WDLT1, ADC_WDHT1, ADC_WDLT2 and ADC_WDHT2 registers.

**Note:** For ADC0/1, if OVSEN = 1, analog watchdog 0/1/2 can compare the analog voltage converted (after oversample) with a low threshold or a high threshold. If OVSEN = 0, analog watchdog 0/1/2 can compare the analog voltage converted (before oversample) with a low threshold or a high threshold.

## 20.4.8. Data storage mode

The alignment of data stored after conversion can be specified by DAL bit in the ADC_CTL1 register.

**Figure 20-7. 14-bit Data storage mode**

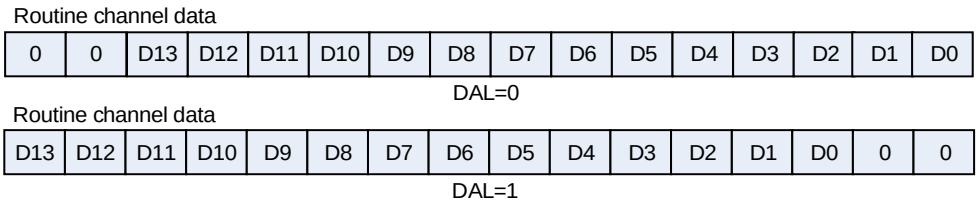
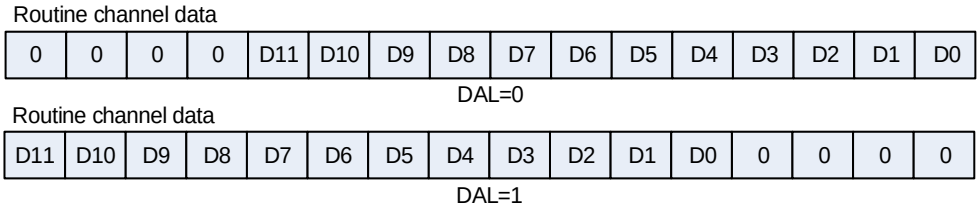
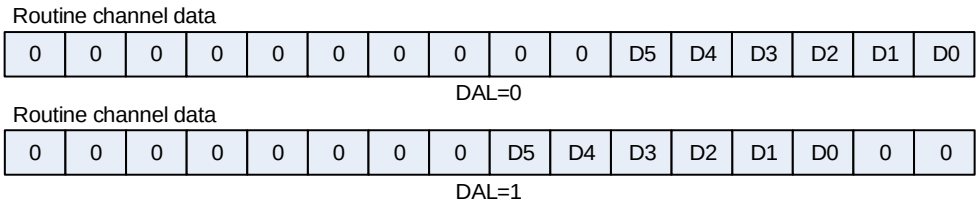



**Figure 20-8. 12-bit Data storage mode**




6-bit resolution data storage mode is different from 14-bit/12-bit/10-bit/8-bit resolution data storage mode, shown as ***Figure 20-9. 6-bit data storage mode***.

**Figure 20-9. 6-bit data storage mode**




**NOTE:** When OVSEN bit in the ADC_OVSAMPCTL register is set, the DAL bit value in the ADC_CTL1 register is ignored and the ADC only support LSB storage mode.

### 20.4.9. Sample time configuration

The number of CK_ADC cycles which is used to sample the input voltage can be specified by the RSMPn[9:0] bits in the ADC_RSQ0~ ADC_RSQ8 registers or ISMPn[9:0] bits in the ADC_ISQ0~ ADC_ISQ2. A different sample time can be specified for each sequence. For 12-bits resolution, the total sampling and conversion time is “sampling time + 12.5” CK_ADC cycles.

Example:

CK_ADC = 40MHz and sample time is 3.5 cycles, the total conversion time is “3.5+12.5” CK_ADC cycles, that means 0.4 us.

### 20.4.10. External trigger configuration

The conversion of routine sequence can be triggered by rising edge of TRIGSEL or software. The trigger source of routine sequence is controlled by the ETMRC[1:0] bits in the ADC_CTL1 register.

**Table 20-4. Trigger source for routine channels for ADC0/ADC1/ADC2**

| ETMRC[1:0] | Trigger Source | Trigger Type |
|---|---|---|
| 01, 10, 11 | TRIGSEL | Signal from TRIGSEL |
| 00 | SWRCST | Software trigger |

### 20.4.11. DMA request

The DMA request, which is enabled by the DMA bit of ADC_CTL1 register, is used to transfer data of routine sequence for conversion of more than one channel. The ADC generates a DMA request at the end of conversion of a routine channel. When this request is received, the DMA will transfer the converted data from the ADC_RDATA register to the destination location which is specified by the user.

### 20.4.12. Overflow detection

Overflow detection is enabled when DMA is enabled or EOCM bit in ADC_CTL1 is set. An overflow event occurs when a routine conversion is done before the prior routine data has

been read out. The ROVF bit of the ADC_STAT is set. Overflow interrupt is generated if the ROVFIE bit in the ADC_CTL0 is set.

It is recommended to reinitialize the DMA module to recover the ADC from ROVF state. To ensure the routine converted data are transferred correctly, the internal state machine is reset. The ADC conversion will be stalled until the ROVF bit is cleared.

Software procedure for recovering the ADC from ROVF state:

1. Clear DMA bit of ADC_CTL1 to 0.
2. Clear ADCON bit of ADC_CTL1 to 0.
3. Clear CHEN bit of DMA_CHxCTL to 0 with reinit DMA module.
4. Clear ROVF bit of ADC_STAT to 0.
5. Set CHEN bit of DMA_CHxCTL to 1.
6. Set DMA bit of ADC_CTL1 to 1.
7. Set ADCON bit of ADC_CTL1 to 1.
8. Wait T(setup).
9. Start conversion with software or trigger.

### 20.4.13. ADC internal channels

When the TSVEN1 bit of ADC_CTL1 register is set, the temperature sensor channel (ADC2_CH18) is enabled. When the TSVEN2 bit of ADC_CTL1 register is set, the high-precison temperature sensor channel (ADC2_CH20) is enabled when the INREFEN bit of ADC_CTL1 register is set, the VREFINT channel (ADC1_CH17/ADC2_CH19) is enabled. The temperature sensor can be used to measure the ambient temperature of the device. The sensor output voltage can be converted into a digital value by ADC. The sampling time for the temperature sensor is recommended to be set to at least $t_{s_temp}$ μs (please refer to the datasheet). When this sensor is not in use, it can be put in power down mode by resetting the TSVEN1 or TSVEN2 bit.

The output voltage of the temperature sensor (only for normal temperature sensor) changes linearly with temperature. Because there is an offset, which is up to 45 °C and varies from chip to chip due to the chip production variation, the internal temperature sensor is more appropriate to detect temperature variations instead of absolute temperature. When it is used to detect accurate temperature, an external temperature sensor part should be used to calibrate the offset error.

The internal voltage reference ($V_{REFINT}$) provides a stable (bandgap) voltage output for the ADC and Comparators. $V_{REFINT}$ is internally connected to the ADC1_CH17/ADC2_CH19 input channel.

To use the temperature sensor:

1. Configure the conversion sequence (ADC2_IN18) and the sampling time($t_{s_temp}$ μs) for the channel.
2. Enable the temperature sensor by setting the TSVEN1 bit in the ADC control register 1 (ADC_CTL1).
3. Start the ADC conversion by setting the ADCON bit or by the triggers.

4. Read the temperature data($V_{temperature}$) in the ADC data register, and get the temperature with the following equation.

   Temperature (°C) = {($V_{25}$ – $V_{temperature}$) / Avg_Slope} + 25.

   $V_{25}$: internal temperature sensor output voltage at 25°C, the typical value and factory calibration value address please refer to the datasheet (in the temperature sensor characteristics chapter).

   Avg_Slope: Average Slope for curve between Temperature vs. internal temperature sensor output voltage, the typical value please refer to the datasheet (in the temperature sensor characteristics chapter).

To use the high precison temperature sensor:

1. Configure the ADC clock(not greater than 5MHz).
2. Configure the conversion sequence (ADC2_CH20) and the sampling time($t_{s_temp}$ μs) for the channel.
3. Enable the temperature sensor by setting the TSVEN2 bit in the ADC control register 1 (ADC_CTL1).
4. Start the ADC conversion by setting the ADCON bit or by the triggers.
5. Read the temperature data($V_{temperature}$) in the ADC data register, and get the temperature with the following equation.

   Temperature (°C) = {($V_{temperature}$ – $V_{25}$) / Avg_Slope} + 25.

   $V_{25}$: internal temperature sensor output voltage at 25°C, the typical value and factory calibration value address please refer to the datasheet (in the high-precision temperature sensor characteristics chapter).

   Avg_Slope: Average Slope for curve between Temperature vs. Vtemperature, the typical value please refer to the datasheet (in the high-precision temperature sensor characteristics chapter).

**Note**:

1) After the high precison temperature sensor is enabled, it is necessary to wait for at least 3 ADC sampling cycles before the ADC conversion code value is considered valid, and the first 3 conversion data should be discarded;
2) The sampling accuracy of high precison temperature sensor can be improved by means of hardware on chip over sampling or software averaging.

### 20.4.14. Battery voltage monitoring

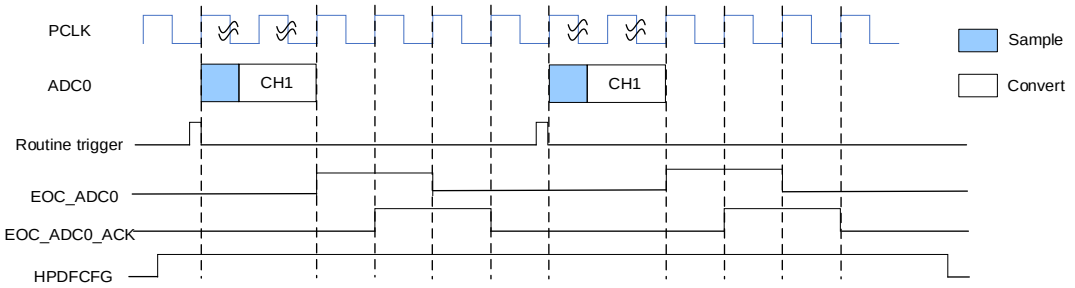
The $V_{BAT}$ channel can be used to measure the backup battery voltage on the $V_{BAT}$ pin. When the VBATEN bit in ADC_CTL1 register is set, $V_{BAT}$ channel (ADC1_IN16 / ADC2_IN17) is enabled and a bridge divider by 4 integrated on the $V_{BAT}$ pin is also enabled automatically with it. As $V_{BAT}$ may be higher than $V_{DDA}$, this bridge is used to ensure the ADC correct operation. And it connects $V_{BAT}$ / 4 to the ADC1_IN16 / ADC2_IN17 input channel. So, the converted digital value is $V_{BAT}$ / 4. In order to prevent unnecessary battery energy consumption, it is

recommended that the bridge will be enabled only when it is required.

### 20.4.15. Using HPDF to managing the conversion results

High-Performance Digital Filter (HPDF)can be used to manage the ADC conversion results. In this situation, The HPDFCFG bit must be set to 1 and DMA bit must be cleared to 0. The ADC transfers 16 least significant bits of the routine data register data to the HPDF, which in turns will reset the EOC flag once the transfer is complete. As shown in ***Figure 20-10. Schematic diagram of handshake signal between HPDF and ADC module***.

**Figure 20-10. Schematic diagram of handshake signal between HPDF and ADC module**




### 20.4.16. Programmable resolution (DRES)

The resolution can be configured to be either 14, 12, 10, 8, or 6 bits by programming the DRES[1:0] bits in the ADC_CTL0 register. For applications that do not require high data accuracy, lower resolution allows faster conversion time. The DRES[1:0] bits must only be changed when the ADCON bit is reset. Lower resolution reduces the conversion time needed for the successive approximation steps as shown in ***Table 20-5. tCONV timings depending on resolution for ADC0 and ADC1***, ***Table 20-6. tCONV timings depending on resolution for ADC2***.

**Table 20-5. $t_{CONV}$ timings depending on resolution for ADC0 and ADC1**

| DRES[1:0] bits | $t_{CONV}$ (ADC clock cycles) | $t_{CONV}$(ns) at $f_{ADC}$=72MHz | $t_{SMPL}$(min) (ADC clock cycles) | $t_{ADC}$ (ADC clock cycles) | $t_{ADC}$(ns) at $f_{ADC}$=72MHz |
|---|---|---|---|---|---|
| 14 | 14.5 | 201.39 ns | 3.5 | 18 | 250 ns |
| 12 | 12.5 | 173.61 ns | 3.5 | 16 | 222.22 ns |
| 10 | 10.5 | 145.83 ns | 3.5 | 14 | 194.44 ns |
| 8 | 8.5 | 118.06 ns | 3.5 | 12 | 166.67 ns |

**Table 20-6. $t_{CONV}$ timings depending on resolution for ADC2**

| DRES[1:0] bits | $t_{CONV}$ (ADC clock cycles) | $t_{CONV}$(ns) at $f_{ADC}$=80MHz | $t_{SMPL}$(min) (ADC clock cycles) | $t_{ADC}$ (ADC clock cycles) | $t_{ADC}$(ns) at $f_{ADC}$=80MHz |
|---|---|---|---|---|---|
| 12 | 12.5 | 156.25 ns | 2.5 | 15 | 187.5 ns |
| 10 | 10.5 | 121.25 ns | 2.5 | 13 | 162.5 ns |
| 8 | 8.5 | 106.25ns | 2.5 | 11 | 137.5ns |

| | | | | | |
|---|---|---|---|---|---|
| 6 | 6.5 | 81.25 ns | 2.5 | 9 | 112.5ns |

### 20.4.17. On-chip hardware oversampling

The on-chip hardware oversampling circuit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 32-bit in ADC0 and ADC1, up to 16-bit in ADC2.The on-chip hardware oversampling circuit is enabled by OVSEN bit in the ADC_OVSAMPCTL register. It provides a result with the following form, where N and M can be adjusted, and Dout(n) is the n-th output digital signal of the ADC:

$$Result=\frac{1}{M}*\sum_{n=0}^{N-1}D_{out}(n) \quad (20\text{-}2)$$

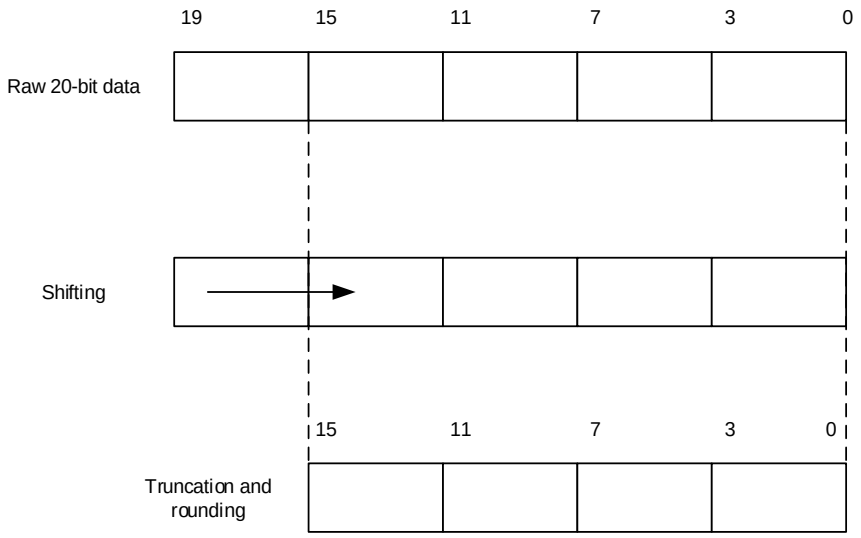
For 14bit-ADC, the on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting. The oversampling ratio N is defined by the OVSR[9:0] bits in the ADC_OVSAMPCTL register. It can range from 2x to1024x. The division coefficient M means bit right shifting up to 11-bit. It is configured through the OVSS[3:0] bits in the ADC_OVSAMPCTL register.

For 14bit-ADC, summation units can produce up to 24 bits (1024 x 14bit), which is first shifted right. Then store the data into register.

For 12bit-ADC, the on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting. The oversampling ratio N is defined by the OVSR[7:0] bits in the ADC_OVSAMPCTL register. It can range from 2x to 256x. The division coefficient M means bit right shifting up to 8-bit. It is configured through the OVSS[3:0] bits in the ADC_OVSAMPCTL register.

For 12bit-ADC, summation units can produce up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the data register.

**Figure 20-11. 20-bit to 16-bit result truncation (for 12bit ADC)**




**Note:** If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

*Figure 20-11. 20-bit to 16-bit result truncation (for 12bit ADC)* shows a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 20-12. Numerical example with 5-bits shift and rounding (for 12bit ADC)**

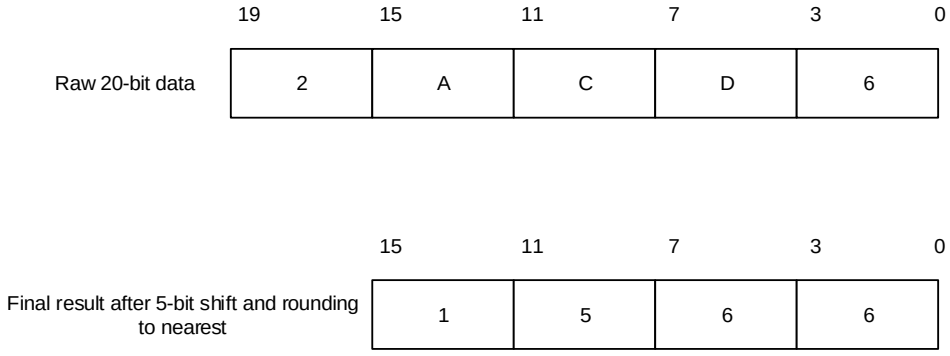



**Figure 20-13. 14bit ADC oversampling with 10bits right shift**

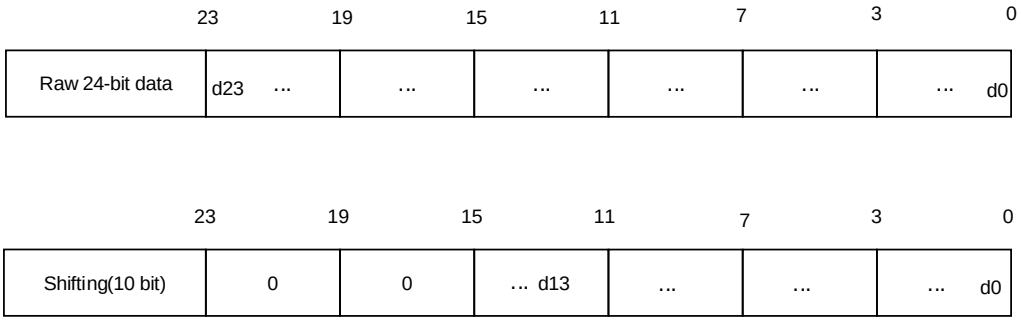
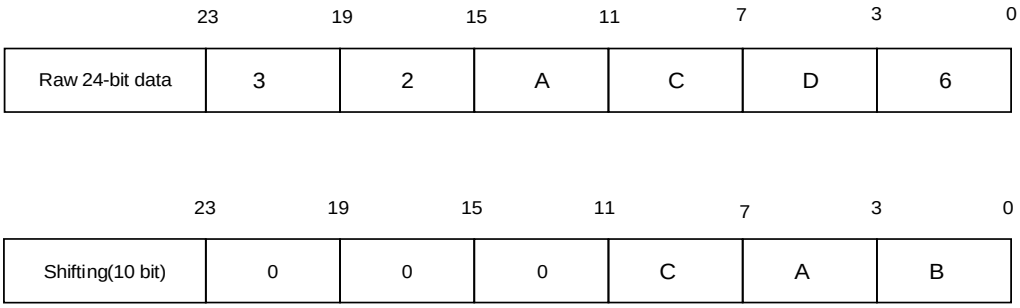


**Figure 20-14. Numerical example for 14bit ADC oversampling with 10bits right shift**




The ***Table 20-7. Some examples show the maximum output results for N and M combimations (grayed values indicates truncation，for 12bit ADC)*** below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

**Table 20-7. Some examples show the maximum output results for N and M combimations (grayed values indicates truncation，for 12bit ADC)**

| Oversampling ratio | Max Raw data | No-shift OVSS=0000 | 1-bit shift OVSS=0001 | 2-bit shift OVSS=0010 | 3-bit shift OVSS=0011 | 4-bit shift OVSS=0100 | 5-bit shift OVSS=0101 | 6-bit shift OVSS=0110 | 7-bit shift OVSS=0111 | 8-bit shift OVSS=1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| 2x | 0x1FFE | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF | 0x007F | 0x003F | 0x001F |
| 4x | 0x3FFC | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF | 0x007F | 0x003F |
| 8x | 0x7FF8 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF | 0x007F |
| 16x | 0xFFF0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF | 0x00FF |
| 32x | 0x1FFE0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF | 0x01FF |
| 64x | 0x3FFC0 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF | 0x03FF |
| 128x | 0x7FF80 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF | 0x07FF |
| 256x | 0xFFF00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE | 0x0FFF |
| 512x | 0x1FFE00 | 0xFE00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC | 0x1FFE |
| 1024x | 0x3FFC00 | 0xFC00 | 0xFE00 | 0xFF00 | 0xFF80 | 0xFFC0 | 0xFFE0 | 0xFFF0 | 0x7FF8 | 0x3FFC |

When compared to standard conversion mode, the conversion timings of oversampling mode do not change, and the sampling time is maintained the same as that of standard conversion mode during the whole oversampling sequence. New data is supplied every N conversions, and the equivalent delay is equal to:

$$N \times t_{ADC} = N \times (t_{SMPL} + t_{CONV}) \quad (20\text{-}3)$$

## 20.5. ADC sync mode

In devices with two ADCs, the ADC sync mode can be used.

In ADC sync mode, the conversion of ADC1 are synchronized by the triggers of ADC0. The two ADCs convert parallelly or rotately, according to the mode selected by the SYNCM[3:0] bits in ADC_SYNCCTL register.

In ADC sync mode, when the conversion is configured to be triggered by an external event, the external trigger must be disabled for ADC1. The converted data of routine sequence is stored in the ADC sync routine data register (ADC_SYNCDATA0 or ADC_SYNCDATA1).

The following modes can be configured in ***Table 20-8. ADC sync mode table***.

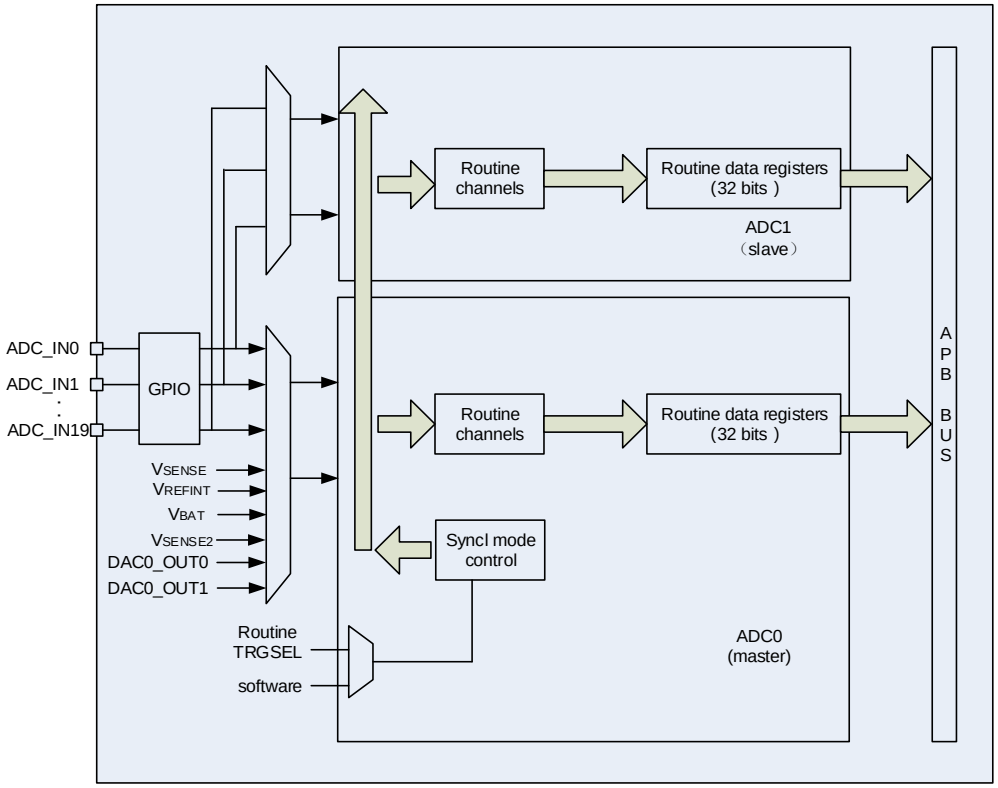
**Table 20-8. ADC sync mode table**

| SYNCM[3: 0] | mode |
|---|---|
| 0000 | Free mode |
| 0110 | Routine parallel mode |
| 0111 | Routine follow-up mode |

When the ADCs are in a sync mode other than free mode, they shold be configured to free mode before being configured to another sync mode.

The ADC sync scheme is shown in ***Figure 20-15. ADC sync block diagram*** .

**Figure 20-15. ADC sync block diagram**




## 20.5.1. Free mode

In this mode, the ADC synchronization is bypassed, and each ADC works freely.

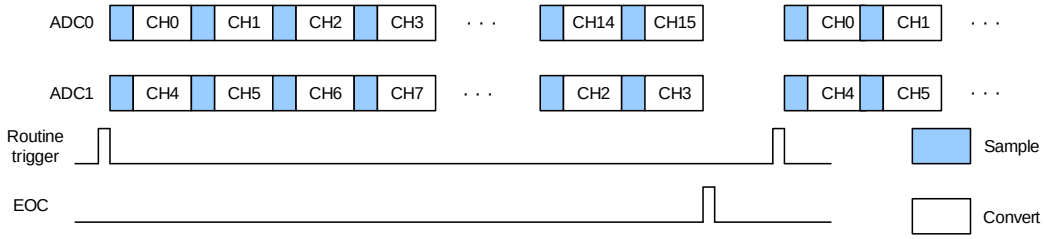
## 20.5.2. Routine parallel mode

The routine parallel mode is enabled by setting the SYNCM[3:0] bits in the ADC_SYNCCTL register to 0110. In the routine parallel mode, all of the ADCs convert the routine sequence at the selected external trigger of ADC0. The triggers is selected by configuring the ETMRC[1:0]

bits in the ADC_CTL1 register of ADC0.

EOC interrupts (if enabled on the ADC interfaces) are generated at the end of conversion events according to the EOCM bit in the ADC_CTL1 register. The behavior of routine parallel mode is shown in the ***Figure 20-16. Routine parallel mode on 16 channels*** .

**Figure 20-16. Routine parallel mode on 16 channels**




**Note:**

1. Do not convert the same channel on two ADCs at a given time (no overlapping sampling times for the ADCs when converting the same channel).
2. Make sure to trigger the ADCs when none of them is converting (do not trigger ADC0 when some of the conversions are not finished).

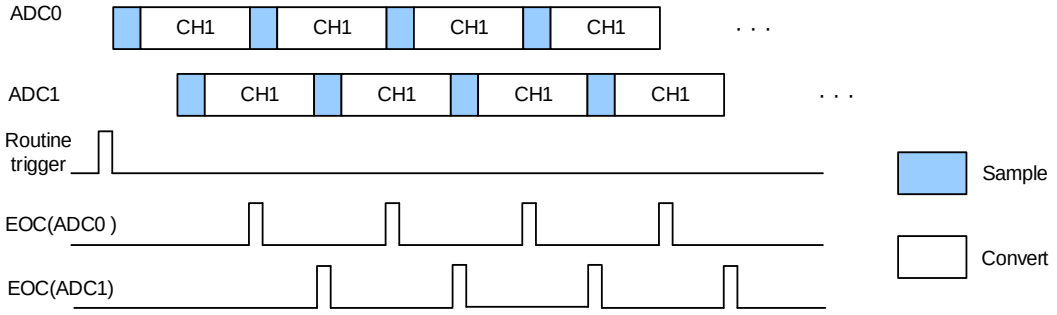
### 20.5.3. Routine follow-up mode

The routine follow-up mode is enabled by setting the SYNCM[3:0] bits in the ADC_SYNCCTL register to 0111. In the follow-up mode, ADC0 converts the routine channel at the selected external trigger. The triggers are selected by configuring the ETMRC[1:0] bits in the ADC_CTL1 register of ADC0. After a delay time, ADC1 converts the routine channel. The routine channel in above descriptions only includes one routine channel.

The delay time between two consecutive sample phase is configured by the SYNCDLY[3:0] bits in the ADC_SYNCCTL register. To prevent more than one ADCs from sampling the same channel at a given time, if the delay time configured by the SYNCDLY bits is shorter than the sample time, the delay time of (sample time + 2) CK_ADC cycles will be used.

If the CNT bit in ADC_CTL1 register is set, the selected routine channels are continuously converted. EOC interrupts (if enabled on the ADC interfaces) are generated at the end of conversion events according to the EOCM bit in the ADC_CTL1 register. The behavior of follow-up mode is shown in the ***Figure 20-17. Routine follow-up mode on 1 channel in continuous operation*** **mode** .

**Figure 20-17. Routine follow-up mode on 1 channel in continuous operation mode**




**Note:**

1. Do not convert the same channel on two ADCs at a given time (no overlapping sampling times for the two ADCs when converting the same channel).
2. Make sure to trigger the ADCs when none of them is converting (do not trigger ADC0 when some of the conversions are not finished).

### 20.5.4. Use DMA in ADC sync mode

In ADC sync mode, the converted data of routine channels are stored in the ADC sync routine data register (ADC_SYNCDATA0 or ADC_SYNCDATA1). DMA can be used to transfer data from ADC_SYNCDATA0 or ADC_SYNCDATA1 register. There are two DMA work modes, which can work well with the various ADC sync modes.

#### ADC sync DMA mode 0

In ADC sync DMA mode 0, the bitwidth of DMA transfer is 32. One DMA request transfers one data, which is selected from the routine data of the ADCs in turn. For every request, the source address of the DMA channel should be fixed to the ADC_SYNCDATA1 register, while the content of the ADC_SYNCDATA changes to the data that is to be transferred. When ADC0 and ADC1 work in SYNC mode, the transfer sequence is ADC0_RDATA[31:0] -> ADC1_RDATA[31:0] -> ADC0_RDATA[31:0] -> ADC1_RDATA[31:0].

The ADC Sync DMA mode 0 is properly for:

- ADC0 and ADC1 work in routine parallel mode (SYNCM=0110).

#### ADC sync DMA mode 1

In ADC sync DMA mode 1, the bitwidth of DMA transfer is 32. One DMA request transfers two data, which are selected from the routine data of the ADCs in turn. For every request, the source address of the DMA channel should be fixed to the ADC_SYNCDATA0 register, while the content of the ADC_SYNCDATA changes to the data that is to be transferred. When ADC0 and ADC1 works in SYNC mode, the transfer data are always {ADC1_RDATA[15:0], ADC0_RDATA[15:0]}.

The ADC Sync DMA mode 1 is properly for:

- ADC0 and ADC1 work in routine parallel mode (SYNCM=0110).
- ADC0 and ADC1 work in routine follow-up mode (SYNCM=0111).

## 20.6. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for routine sequence.
- The analog watchdog event.
- Overflow event.

The interrupts of ADC0, ADC1 and ADC2 are mapped into the same interrupt vector IRQ18.

## 20.7. Register definition

ADC0 base address: 0x4001 2400

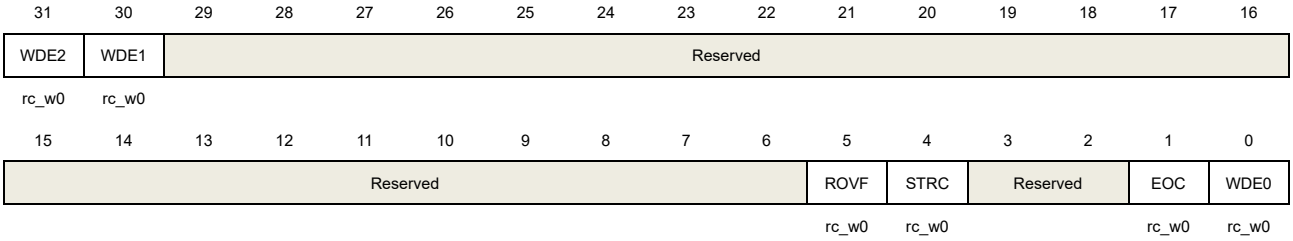
ADC1 base address: 0x4001 2800

ADC2 base address: 0x4001 2C00

### 20.7.1. Status register (ADC_STAT)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | WDE2 | Analog watchdog 2 event flag<br>0: No analog watchdog 2 event<br>1: Analog watchdog 2 event<br>Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT2 and ADC_WDLT2 register. Cleared by software writing 0 to it. |
| 30 | WDE1 | Analog watchdog 1 event flag<br>0: No analog watchdog 1 event<br>1: Analog watchdog 1 event<br>Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT1 and ADC_WDLT1 register. Cleared by software writing 0 to it. |
| 29:6 | Reserved | Must be kept at reset value. |
| 5 | ROVF | Routine sequence data register overflow<br>0: Routine sequence data register not overflow<br>1: Routine sequence data register overflow<br>This bit is set by hardware when the routine sequence data registers are overflow, in single mode or multi mode. This flag is only set when DMA is enabled or end of conversion mode is set to 1(EOCM=1). The recent routine sequence data is lost when this bit is set.<br>Cleared by software writing 0 to it. |
| 4 | STRC | Start flag of routine sequence conversion<br>0: Conversion is not started |

| | | |
|---|---|---|
| | | 1: Conversion is started<br>Set by hardware when routine sequence conversion starts. Cleared by software writing 0 to it. |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | EOC | End flag of routine sequence conversion<br>0: No end of routine sequence conversion<br>1: End ofroutine sequence conversion<br>Set by hardware at the end of a routine sequence conversion.<br>Cleared by software writing 0 to it or by reading the ADC_RDATA register. |
| 0 | WDE0 | Analog watchdog 0 event flag<br>0: No analog watchdog 0 event<br>1: Analog watchdog 0 event<br>Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT and ADC_WDHT registers. Cleared by software writing 0 to it. |

## 20.7.2. Control register 0 (ADC_CTL0)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDE2IE | WDE1IE | Reserved | | | ROVFIE | DRES[1:0] | | RWD0EN | Reserved | | | | | | |
| rw | rw | | | | rw | rw | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISNUM[2:0] | | | Reserved | DISRC | Reserved | WD0SC | SM | Reserved | WDE0IE | EOCIE | WD0CHSEL[4:0] | | | | |
| rw | | | | rw | | rw | rw | | rw | rw | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | WDE2IE | Interrupt enable for WDE2<br>0: WDE2 interrupt disable<br>1: WDE2 interrupt enable |
| 30 | WDE1IE | Interrupt enable for WDE1<br>0: WDE1 interrupt disable<br>1: WDE1 interrupt enable |
| 29:27 | Reserved | Must be kept at reset value. |
| 26 | ROVFIE | Interrupt enable for ROVF<br>0: ROVF interrupt disable<br>1: ROVF interrupt enable |
| 25:24 | DRES[1:0] | ADC data resolution for ADC0/ADC1<br>00: 14bit |

| | | |
|---|---|---|
| | | 01: 12bit<br>10: 10bit<br>11: 8bit<br>ADC data resolution for ADC2<br>00: 12bit<br>01: 10bit<br>10: 8bit<br>11: 6bit |
| 23 | RWD0EN | Routine channel analog watchdog 0 enable<br>0: Routine channel analog watchdog 0 disable<br>1: Routine channel analog watchdog 0 enable |
| 22:16 | Reserved | Must be kept at reset value. |
| 15:13 | DISNUM[2:0] | Number of conversions in discontinuous mode<br>The number of channels to be converted after a trigger will be DISNUM+1 |
| 12 | Reserved | Must be kept at reset value. |
| 11 | DISRC | Discontinuous mode on routine sequence<br>0: Discontinuous operation mode disable<br>1: Discontinuous operation mode enable |
| 10 | Reserved | Must be kept at reset value. |
| 9 | WD0SC | When in scan mode, analog watchdog 0 is effective on a single channel<br>0: Analog watchdog 0 is effective on all channels<br>1: Analog watchdog 0 is effective on a single channel |
| 8 | SM | Scan mode<br>0: Scan operation mode disable<br>1: Scan operation mode enable |
| 7 | Reserved | Must be kept at reset value. |
| 6 | WDE0IE | Interrupt enable for WDE0<br>0: Interrupt disable<br>1: Interrupt enable |
| 5 | EOCIE | Interrupt enable for EOC<br>00: Interrupt disable<br>1: Interrupt enable |
| 4:0 | WD0CHSEL[4:0] | Analog watchdog 0 channel select<br>00000: ADC channel0<br>00001: ADC channel1<br>00010: ADC channel2<br>00011: ADC channel 3<br>00100: ADC channel 4 |

00101: ADC channel 5
00110: ADC channel 6
00111: ADC channel 7
01000: ADC channel 8
01001: ADC channel 9
01010: ADC channel 10
01011: ADC channel 11
01100: ADC channel 12
01101: ADC channel 13
01110: ADC channel 14
01111: ADC channel15
10000: ADC channel16
10001: ADC channel17
10010: ADC channel18
10011: ADC channel19
10100: ADC channel20
Other values are reserved.

**Note:** ADC0 analog inputs Channel20 is internally connected to DAC0_OUT0. ADC1 analog inputs Channel16、Channel17、Channel20 are internally connected to the battery、$V_{REFINT}$ inputs and DAC0_OUT1. ADC2 analog inputs Channel17、Channel18、Channel Channel19、Channel20 are internally connected to the battery、tempeture sensor、$V_{REFINT}$ inputs and high-precision tempeture sensor.

### 20.7.3. Control register 1 (ADC_CTL1)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TSVEN2 | SWRCST | ETMRC[1:0] | | CALMOD | Reserved | VBATEN | INREFEN | TSVEN1 | Reserved | | | | | | |
| rw | rw | rw | | rw | | rw | rw | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | HPDFCFG | DAL | EOCM | DDM | DMA | Reserved | CALNUM[2:0] | | | RSTCLB | CLB | CTN | ADCON |
| | | | rw | rw | rw | rw | rw | | rw | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TSVEN2 | This bit can be set or cleared by software in ADC2.<br>Channel 20(high-precision temperature sensor) enable of ADC2.<br>0: high-precision temperature sensor Channel disable<br>1: high-precision temperature sensor Channel enable |
| 30 | SWRCST | Software start conversion of routine sequence .<br>Setting 1 on this bit starts a conversion of a routine sequence channels. It is set by |

| | | |
|---|---|---|
| | | software and cleared by software or by hardware immediately after the conversion starts. |
| 29:28 | ETMRC[1:0] | External trigger mode for routine sequence<br>00: External trigger for routine sequence disable<br>01: Rising edge of external trigger for routine sequence enable<br>10: Falling edge of external trigger for routine sequence enable<br>11: Rising and falling edge of external trigger for routine sequence enable |
| 27 | CALMOD | ADC calibration mode (for ADC0/1)<br>0: offset、mismatch mode<br>1: offset mode |
| 26 | Reserved | Must be kept at reset value. |
| 25 | VBATEN | This bit can be set or cleared by software in ADC2.<br>Channel 16 (1/4 voltage of external battery) enable of ADC1<br>Channel 17 (1/4 voltage of external battery) enable of ADC2<br>0: 1/4 voltage of external battery Channel disable<br>1: 1/4 voltage of external battery Channel enable |
| 24 | INREFEN | This bit can be set or cleared by software in ADC2.<br>Channel 17 (internal reference voltage) enable of ADC1.<br>Channel 19 (internal reference voltage) enable of ADC2.<br>0: internal reference voltage Channel disable<br>1: internal reference voltage Channel enable |
| 23 | TSVEN1 | This bit can be set or cleared by software in ADC2.<br>Channel 18(temperature sensor) enable of ADC2.<br>0: temperature sensor Channel disable<br>1: temperature sensor Channel enable |
| 22:13 | Reserved | Must be kept at reset value. |
| 12 | HPDFCFG | HPDF mode configuration<br>To enable the HPDF mode, this bit is set and cleared by software. It is only valid when DMA=0.<br>0: HPDF mode disabled<br>1: HPDF mode enabled |
| 11 | DAL | Data alignment<br>0: LSB alignment<br>1: MSB alignment |
| 10 | EOCM | End of conversion mode<br>0: Only at the end of a routine sequence conversions, the EOC bit is set. Overflow detection is disabled unless DMA=1.<br>1: At the end of each routine sequence conversion, the EOC bit is set. Overflow is detected automatically |

| | | |
|---|---|---|
| 9 | DDM | DMA disable mode<br>This bit configure the DMA disable mode for single ADC mode<br>0: The DMA engine is disabled after the end of transfer signal from DMA controller is detected.<br>1: When DMA=1, the DMA engine issues a request at end of each routine sequence conversion. |
| 8 | DMA | DMA request enable.<br>0: DMA request disable<br>1: DMA request enable |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | CALNUM[2:0] | Calibration Times<br>These bits define the calibration times for ADC.<br>000:1 time<br>001:2 times<br>010:4times<br>011:8times<br>100:16times<br>101:32times(only for 12-bit ADC)<br>Others:reserved. |
| 3 | RSTCLB | Reset calibration<br>This bit is set by software and cleared by hardware after the calibration registers are initialized.<br>0: Calibration register initialize done.<br>1: Initialize calibration register start |
| 2 | CLB | ADC calibration<br>0: Calibration done<br>1: Calibration start |
| 1 | CTN | Continuous mode<br>0: Continuous operation mode disable<br>1: Continuous operation mode enable |
| 0 | ADCON | ADC ON. The ADC will be wake up when this bit is changed from low to high and take a stabilization time. For power saving, when this bit is reset, the analog submodule will be put into powerdown mode.<br>0: ADC disable and power down<br>1: ADC enable |

### 20.7.4. Watchdog high threshold register0 (ADC_WDHT0)

Address offset: 0x1C

Reset value: 0x00FF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WDHT0[23:16] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDHT0[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:0 | WDHT0[23:0] | High threshold for analog watchdog 0, For ADC0/ADC1 are WDHT0[23:0], for ADC2 is WDHT0[11:0].<br>These bits define the high threshold for the analog watchdog 0. |

## 20.7.5. Watchdog low threshold register0 (ADC_WDLT0)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WDLT0[23:16] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDLT0[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:0 | WDLT0[23:0] | Low threshold for analog watchdog 0. For ADC0/ADC1 are WDLT0[23:0], for ADC2 is WDLT0[11:0].<br>These bits define the low threshold for the analog watchdog. |

## 20.7.6. Routine sequence register 0 (ADC_RSQ0)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | RL[3:0] | | | | Reserved | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | RSMP15[9:0] | RSQ15[4:0] |
|---|---|---|
| | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:20 | RL[3:0] | Routine channel length.<br>The total number of conversion in routine sequence equals to RL[3:0]+1. |
| 19:15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP15[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ15[4:0] | refer to RSQ0[4:0] description |

## 20.7.7. Routine sequence register 1 (ADC_RSQ1)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP14[9:0] | | | | | | | | | | RSQ14[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP13[9:0] | | | | | | | | | | RSQ13[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP14[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles |

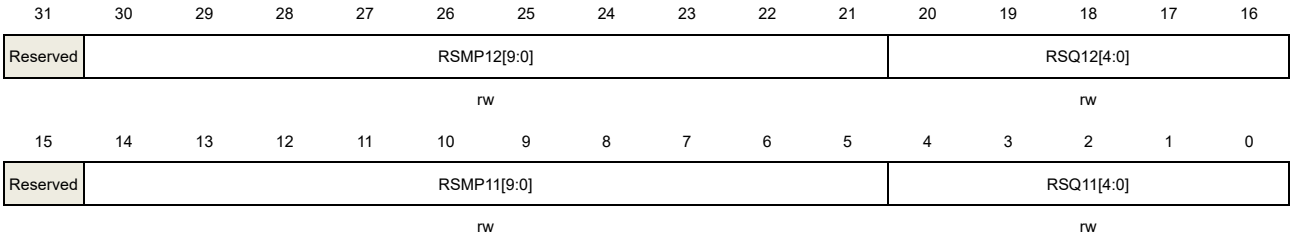
| | | |
|---|---|---|
| | | ……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ14[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP13[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ13[4:0] | refer to RSQ0[4:0] description |

### 20.7.8. Routine sequence register 2 (ADC_RSQ2)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP12[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>…… |

| | | |
|---|---|---|
| | | 10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ12[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP11[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ11[4:0] | refer to RSQ0[4:0] description |

## 20.7.9. Routine sequence register 3 (ADC_RSQ3)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP10[9:0] | | | | | | | | | | RSQ10[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RSMP9[9:0] | | | | | | | | | | RSQ9[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP10[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles |

| | | |
|---|---|---|
| | | 10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ10[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP9[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ9[4:0] | refer to RSQ0[4:0] description |

## 20.7.10. Routine sequence register 4 (ADC_RSQ4)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP8[9:0] | | | | | | | | | | RSQ8[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP7[9:0] | | | | | | | | | | RSQ7[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP8[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles |

| | | |
|---|---|---|
| | | ……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ8[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP7[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ7[4:0] | refer to RSQ0[4:0] description |

## 20.7.11. Routine sequence register 5 (ADC_RSQ5)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP6[9:0] | | | | | | | | | | RSQ6[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP5[9:0] | | | | | | | | | | RSQ5[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP6[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>…… |

| | | |
|---|---|---|
| | | 10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ6[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP5[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ5[4:0] | refer to RSQ0[4:0] description |

## 20.7.12. Routine sequence register 6 (ADC_RSQ6)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP4[9:0] | | | | | | | | | | RSQ4[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP3[9:0] | | | | | | | | | | RSQ3[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP4[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>…… |

| | | |
|---|---|---|
| | | 10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ4[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP3[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ3[4:0] | refer to RSQ0[4:0] description |

## 20.7.13. Routine sequence register 7 (ADC_RSQ7)

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP2[9:0] | | | | | | | | | | RSQ2[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | RSMP1 [9:0] | | | | | | | | | | RSQ1[4:0] | | | | |
| | rw | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | RSMP2[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>…… |

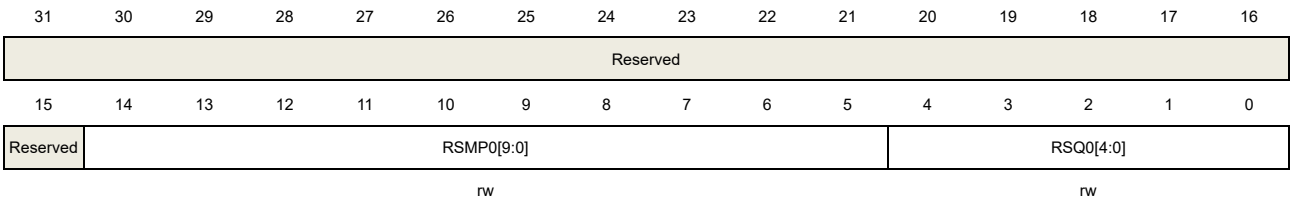
| | | |
|---|---|---|
| | | 10’d807: Only for ADC0/1 is 810.5 cycles |
| 20:16 | RSQ2[4:0] | refer to RSQ0[4:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP1[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |
| 4:0 | RSQ1[4:0] | refer to RSQ0[4:0] description |

## 20.7.14. Routine sequence register 8 (ADC_RSQ8)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



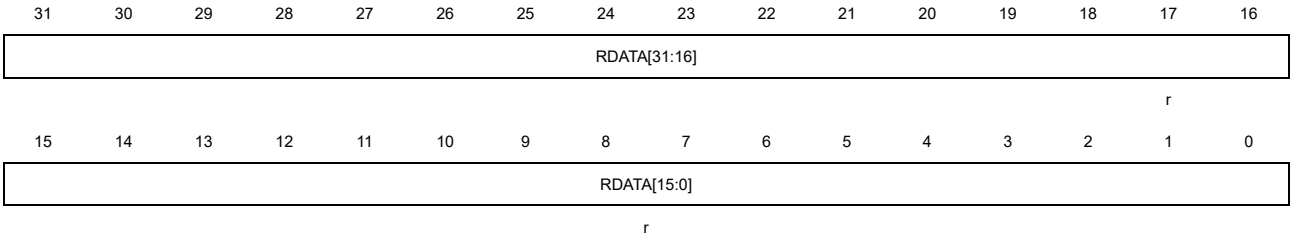

| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14:5 | RSMP0[9:0] | Routine sequence sample time<br>10’d0: For ADC0/1 is 3.5 cycles, For ADC2 is 2.5 cycles<br>10’d1: For ADC0/1 is 4.5 cycles, For ADC2 is 3.5 cycles<br>10’d2: For ADC0/1 is 5.5 cycles, For ADC2 is 4.5 cycles<br>10’d3: For ADC0/1 is 6.5 cycles, For ADC2 is 5.5 cycles<br>10’d4: For ADC0/1 is 7.5 cycles, For ADC2 is 6.5 cycles<br>……<br>10’d638: For ADC0/1 is 641.5 cycles, For ADC2 is 640.5 cycles<br>10’d639: Only for ADC0/1 is 642.5 cycles<br>……<br>10’d807: Only for ADC0/1 is 810.5 cycles |

| | | |
|---|---|---|
| 4:0 | RSQ0[4:0] | The channel number (0..20) is written to these bits to select a channel as the nth conversion in the routine sequence. |

## 20.7.15. Routine data register (ADC_RDATA)

Address offset: 0x64
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

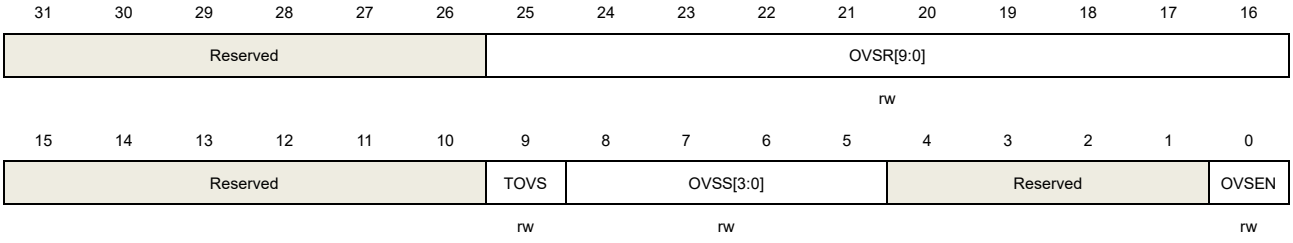



| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RDATA[31:0] | Routinel data. For ADC0/ADC1, RDATA is [31:0], for ADC2, RDATA is [15:0]. These bits contain the conversion result from routine sequence, which is read only. |

## 20.7.16. Oversample control register (ADC_OVSAMPCTL)

Address offset: 0x80
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25:16 | OVSR[9:0] | Oversampling ratio<br>This bit filed defines the number of oversampling ratio. ADC0/1 is 1x~1024x. ADC2 is 1X~256X.<br>10'd0: 1x(no oversampling)<br>10'd1: 2x<br>10'd2: 3x<br>…… |

| | | |
|---|---|---|
| | | 10'd1023:1024x<br><br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |
| 15:10 | Reserved | Must be kept at reset value. |
| 9 | TOVS | Triggered Oversampling<br>This bit is set and cleared by software.<br>0: All oversampled conversions for a channel are done consecutively after a trigger<br>1: Each conversion needs a trigger for a oversampled channel and the number of triggers is determined by the oversampling ratio(OVSR[9:0]).<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |
| 8:5 | OVSS[3:0] | Oversampling shift<br>This bit is set and cleared by software. For ADC0/1, OVSS change from 0000~1101. For ADC2, OVSS change from 0000~1000.<br>0000: No shift<br>0001: Shift 1-bit<br>0010: Shift 2-bits<br>0011: Shift 3-bits<br>0100: Shift 4-bits<br>0101: Shift 5-bits<br>0110: Shift 6-bits<br>0111: Shift 7-bits<br>1000: Shift 8-bits<br>1001: Shift 9-bits<br>1010: Shift 10-bits<br>1011: Shift 11-bits<br>Other codes reserved<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |
| 4:1 | Reserved | Must be kept at reset value. |
| 0 | OVSEN | Oversampler Enable<br>This bit is set and cleared by software.<br>0: Oversampler disabled<br>1: Oversampler enabled<br>**Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress). |

### 20.7.17. Watchdog 1 Channel Selection Register (ADC_WD1SR)

Address offset: 0xA0

Reset value: 0x00000000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | AWD1CS[20:16] | | | | |
| | | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AWD1CS[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:0 | AWD1CS[20:0] | Analog watchdog 1 channel selection<br>These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 1.<br>AWD1CS[n] = 0: ADC analog input channel n is not monitored by analog watchdog 1.<br>AWD1CS[n] = 1: ADC analog input channel n is monitored by analog watchdog 1.<br>When AWD1CS[20:0] = 000..0, the analog Watchdog 1 is disabled<br>**Note:**<br>1) The channels selected by AWD1CS must be also selected into the ADC_RSQn or ADC_ISQ registers.<br>2) Software is allowed to write these bits only when the ADC is disabled (ADCON =0). |

## 20.7.18. Watchdog 2 Channel Selection Register (ADC_WD2SR)

Address offset: 0xA4

Reset value: 0x00000000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | AWD2CS[20:16] | | | | |
| | | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AWD2CS[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:0 | AWD2CS[20:0] | Analog watchdog 2 channel selection<br>These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.<br>AWD2CS[n] = 0: ADC analog input channel n is not monitored by analog watchdog 2. |

AWD2CS[n] = 1: ADC analog input channel n is monitored by Analog watchdog 2.

When AWD2CS[20:0] = 000..0, the analog Watchdog 2 is disabled

**Note:**

1) The channels selected by AWD2CS must be also selected into the ADC_RSQn or ADC_ISQ registers.
2) Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

## 20.7.19. Watchdog high threshold register1 (ADC_WDHT1)

Address offset: 0xA8

Reset value: 0x00FF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WDHT1[23:16] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDHT1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:0 | WDHT1[23:0] | High threshold for analog watchdog 1. For ADC0/1 are WDHT1[23:0], for ADC2 is WDHT1[7:0].<br>These bits define the high threshold for the analog watchdog 1.<br>**Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0). |

## 20.7.20. Watchdog low threshold register1 (ADC_WDLT1)

Address offset: 0xAC

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WDLT1[23:16] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDLT1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

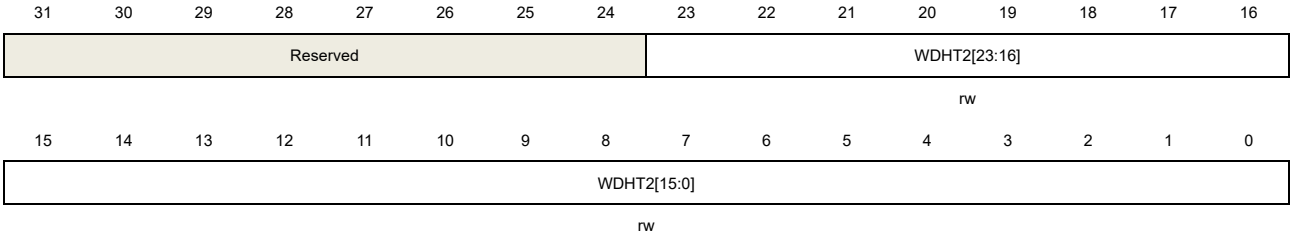
| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 23:0 | WDLT1[23:0] | Low threshold for analog watchdog 1. For ADC0/1 are WDLT1[23:0], for ADC2 is WDLT1[7:0].<br>These bits define the high threshold for the analog watchdog 1.<br>**Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0). |

## 20.7.21. Watchdog high threshold register2 (ADC_WDHT2)

Address offset: 0xB0
Reset value: 0x00FF FFFF

This register has to be accessed by word (32-bit).

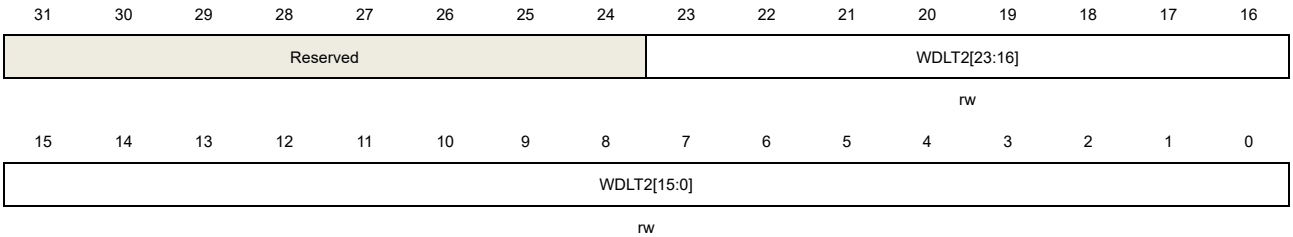



| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:0 | WDHT2[23:0] | High threshold for analog watchdog 2. For ADC0/1 are WDHT2[23:0], for ADC2 is WDHT2[7:0].<br>These bits define the high threshold for the analog watchdog 2.<br>**Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0). |

## 20.7.22. Watchdog low threshold register2 (ADC_WDLT2)

Address offset: 0xB4
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:0 | WDLT2[23:0] | Low threshold for analog watchdog 2. For ADC0/1 are WDLT2[23:0], for ADC2 is |

WDLT2[7:0].

These bits define the high threshold for the analog watchdog 2.

**Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0).

## 20.7.23. Differential mode control register (ADC_DIFCTL)

Address offset: 0XB8

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | DIFCTL[21:16] | | | | | |
| | | | | | | | | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DIFCTL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:0 | DIFCTL[21:0] | Differential mode for channel 21..0.<br>These bits are configured to select whether a channel is in single-ended or differential mode.<br>DIFCTL[i] = 0: ADC analog input channel-i is configured in single-ended mode<br>DIFCTL[i] = 1: ADC analog input channel-i is configured in differential mode<br>**Note:** Software is allowed to write these bits only when the ADC is disabled (ADCON =0). |

## 20.7.24. Summary status register (ADC_SSTAT)

Address offset: 0x300 (for ADC0 base address)

Reset value: 0x0000 0000

This register is read only and provides a summary of the three ADCs. This register is not available in ADC1 and ADC2.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | ADC2_ROVF | ADC2_STRC | Reserved | | ADC2_EOC | ADC2_WDE2 | ADC2_WDE1 | ADC2_WDE0 |
| | | | | | | | | r | r | | | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADC1_ROVF | ADC1_STRC | Reserved | | ADC1_EOC | ADC1_WDE2 | ADC1_WDE1 | ADC1_WDE0 | ADC0_ROVF | ADC0_STRC | Reserved | | ADC0_EOC | ADC0_WDE2 | ADC0_WDE1 | ADC0_WDE0 |

r r r r r r r r r r r r r r r r

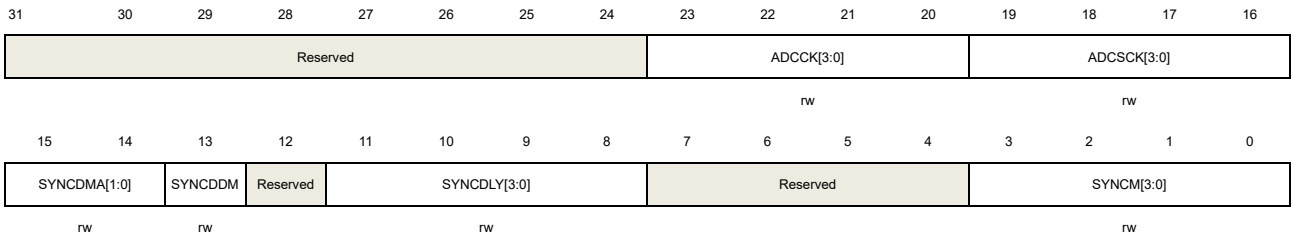
| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | ADC2_ROVF | This bit is the mirror image of the ROVF bit of ADC2 |
| 22 | ADC2_STRC | This bit is the mirror image of the STRC bit of ADC2 |
| 21:20 | Reserved | Must be kept at reset value. |
| 19 | ADC2_EOC | This bit is the mirror image of the EOC bit of ADC2 |
| 18 | ADC2_WDE2 | This bit is the mirror image of the WDE2 bit of ADC2 |
| 17 | ADC2_WDE1 | This bit is the mirror image of the WDE1 bit of ADC2 |
| 16 | ADC2_WDE0 | This bit is the mirror image of the WDE0 bit of ADC2 |
| 15 | ADC1_ROVF | This bit is the mirror image of the ROVF bit of ADC1 |
| 14 | ADC1_STRC | This bit is the mirror image of the STRC bit of ADC1 |
| 13:12 | Reserved | Must be kept at reset value. |
| 11 | ADC1_EOC | This bit is the mirror image of the EOC bit of ADC1 |
| 10 | ADC1_WDE2 | This bit is the mirror image of the WDE2 bit of ADC1 |
| 9 | ADC1_WDE1 | This bit is the mirror image of the WDE1 bit of ADC1 |
| 8 | ADC1_WDE0 | This bit is the mirror image of the WDE0 bit of ADC1 |
| 7 | ADC0_ROVF | This bit is the mirror image of the ROVF bit of ADC0 |
| 6 | ADC0_STRC | This bit is the mirror image of the STRC bit of ADC0 |
| 5:4 | Reserved | Must be kept at reset value. |
| 3 | ADC0_EOC | This bit is the mirror image of the EOC bit of ADC0 |
| 2 | ADC0_WDE2 | This bit is the mirror image of the WDE2 bit of ADC0 |
| 1 | ADC0_WDE1 | This bit is the mirror image of the WDE1 bit of ADC0 |
| 0 | ADC0_WDE0 | This bit is the mirror image of the WDE0 bit of ADC0 |

### 20.7.25. Sync control register (ADC_SYNCCTL)

Address offset: 0x304 (for ADC0 / ADC2 base address)
Reset value: 0x0000 0000

This register is not available in ADC1.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:20 | ADCCK[3:0] | ADC clock prescaler.<br>ADC Prescaler<br>These bits are set and cleared by by software to select the frequency of the ADC clock. All ADCs are common.<br>4'b0000: ADC clock div1<br>4'b0001: ADC clock div2<br>4'b0010: ADC clock div4<br>4'b0011: ADC clock div6<br>4'b0100: ADC clock div8<br>4'b0101: ADC clock div10<br>4'b0110: ADC clock div12<br>4'b0111: ADC clock div16<br>4'b1000: ADC clock div32<br>4'b1001: ADC clock div64<br>4'b1010: ADC clock div128<br>4'b1011: ADC clock div256<br>All other values are reserved. |
| 19:16 | ADCSCK[3:0] | ADC sync clock mode<br>These bits are set and cleared by by software to define the ADC sync clock mode. All ADCs are common.<br>4'b0000: CLK_ADC(async clock mode)<br>4'b1000:HCLK div2(sync clock mode)<br>4'b1001:HCLK div4(sync clock mode)<br>4'b1010:HCLK div6(sync clock mode)<br>4'b1011:HCLK div8(sync clock mode)<br>4'b1100:HCLK div10(sync clock mode)<br>4'b1101:HCLK div12(sync clock mode)<br>4'b1110:HCLK div14(sync clock mode)<br>4'b1111:HCLK div16(sync clock mode)<br>All other values are reserved. |
| 15:14 | SYNCDMA[1:0] | ADC sync DMA mode selection<br>00: ADC sync DMA disabled<br>01: ADC sync DMA mode 0<br>10: ADC sync DMA mode 1 |

| | | |
|---|---|---|
| | | 11: reserved |
| 13 | SYNCDDM | ADC sync DMA disable mode<br>This bit configures the DMA disable mode for ADC sync mode<br>0: The DMA engine is disabled after the end of transfer signal from DMA controller is detected.<br>1: When SYNCDMA is not equal to 2'b00, the DMA engine issues requests according to the SYNCDMA bits. |
| 12 | Reserved | Must be kept at reset value. |
| 11:8 | SYNCDLY[3:0] | ADC sync delay<br>These bits are used to configure the delay between 2 sampling phases in ADC sync modes to (5+SYNCDLY) ADC clock cycles. |
| 7:4 | Reserved | Must be kept at reset value. |
| 3:0 | SYNCM[3:0] | ADC sync mode<br>When ADC sync mode is enabled these bits should be set to 0000 firstly before change to another value.<br>0000: ADC sync mode disabled. All the ADCs work independently.<br>0110: ADC0 and ADC1 work in routine parallel mode.<br>0111: ADC0 and ADC1 work in follow-up mode.<br>All other values are reserved. |

### 20.7.26. Sync routine data register0 (ADC_SYNCDATA0)

Address offset: 0x308 (for ADC0 base address)
Reset value: 0x0000 0000

This register is not available in ADC1 and ADC2.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYNCDATA1[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYNCDATA0[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

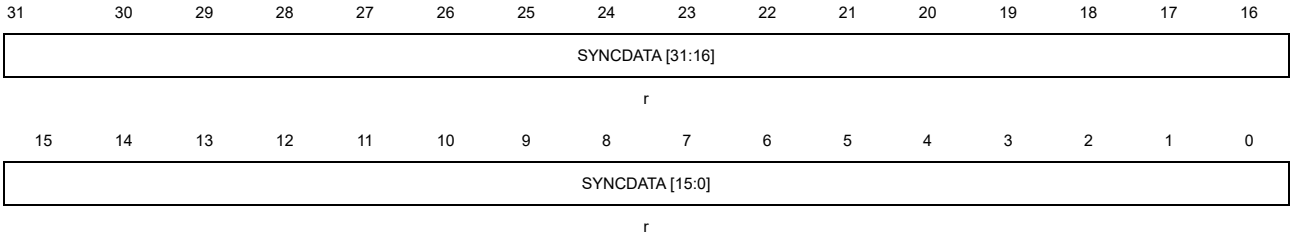
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | SYNCDATA1[15:0] | Routine data1(slave adc routine data) in ADC sync mode. SYNCDMA[1:0] must be 2'b10. |
| 15:0 | SYNCDATA0[15:0] | Routine data0 (master adc routine data) in ADC sync mode. SYNCDMA[1:0] must be 2'b10, |

### 20.7.27. Sync routine data register1 (ADC_SYNCDATA1)

Address offset: 0x30C (for ADC0 base address)
Reset value: 0x0000 0000

This register is not available in ADC1 and ADC2.

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SYNCDATA[31:0] | SYNCDMA[1:0]=2'b01, which is selected from the routine data (master / slave) of the ADCs in turn. |

# 21. Digital-to-analog converter (DAC)

## 21.1. Overview

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The digital data can be configured to 8-bit or 12-bit mode, left-aligned or right-aligned mode. DMA can be used to update the digital data on external triggers.

The output voltage can be optionally buffered for higher drive capability, and DAC output buffer can be calibrated to improve output accuracy.

The sample and keep mode can reduce the power consumption of DAC.

The DAC channels can work independently or concurrently.

## 21.2. Characteristics

The main features of DAC are as follows:

- 8-bit or 12-bit resolution.
- Left or right data alignment.
- DMA capability for each channel and underrun function.
- Conversion update synchronously.
- Conversion triggered by external triggers.
- Configurable internal buffer.
- Extern voltage reference, $V_{REFP}$.
- Output buffer calibration.
- Using sample and keep mode to reduce the power consumption.
- Noise wave generation (LFSR noise mode and Triangle noise mode).
- Two DAC channels in concurrent mode.

*Figure 21-1. DAC block diagram* and *Table 21-1. DAC I/O description* show the block diagram of DAC and the pin description of DAC, respectively.

**Figure 21-1. DAC block diagram**

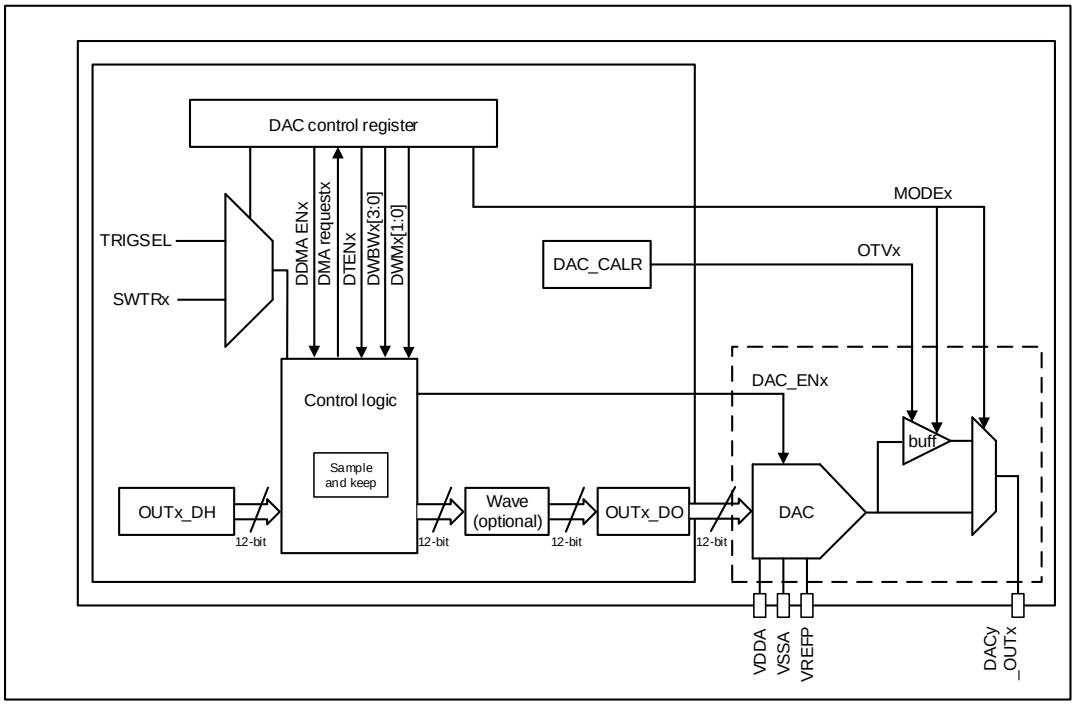



**Table 21-1. DAC I/O description**

| Name | Description | Signal type |
|---|---|---|
| $V_{DDA}$ | Analog power supply | Input, analog supply |
| $V_{SSA}$ | Ground for analog power supply | Input, analog supply ground |
| $V_{REFP}$ | Positive reference voltage of DAC | Input, analog positive reference |
| DACy_OUTx | DAC analog output | Analog output signal |

The below table details the triggers and outputs of the DAC.

**Table 21-2. DAC triggers and outputs summary**

| | DAC0 | |
|---|---|---|
| **Channel** | **Channel0** | **Channel1** |
| **DAC outputs connected to I / Os** | PA4 | PA5 |
| **DAC output buffer** | • | • |
| **DAC trigger signals from TRIGSEL** | • | |
| **DAC software trigger** | • | |

**Note:** The GPIO pins should be configured to analog mode before enable the DAC module.

## 21.3. Function description

### 21.3.1. DAC enable

The DAC can be turned on by setting the DENx bit in the DAC_CTL0 register. A $t_{WAKEUP}$ time is needed to startup the analog DAC submodule.

### 21.3.2. DAC output buffer

For reducing output impedance and driving external loads without an external operational amplifier, an output buffer is integrated inside each DAC module.

The output buffer, which is turned on by default to reduce the output impedance and improve the driving capability, can be turned off by setting the DBOFFx bit in the DAC_CTL0 register.

### 21.3.3. DAC data configuration

The 12-bit DAC holding data (OUTx_DH) can be configured by writing any one of the OUTx_R12DH, OUTx_L12DH and OUTx_R8DH registers. When the data is loaded by OUTx_R8DH register, only the MSB 8 bits are configurable, the LSB 4 bits are forced to 4'b0000.

### 21.3.4. DAC trigger

The DAC conversion can be triggered by software or rising edge of external trigger source. The DAC external trigger is enabled by setting the DTENx bits in the DAC_CTL0 register. The DAC external triggers are selected by the DTSELx bits in the DAC_CTL0 register, which is shown as ***Table 21-3. Triggers of DAC***.

**Table 21-3. Triggers of DAC**

| DTSELx[1:0] | Trigger Source | Trigger Type |
|---|---|---|
| 2b'00 | TRIGSEL | Hardware trigger |
| 2b'01 | SWTR | Software trigger |
| 2b'10 | Reserved | Reserved |
| 2b'11 | | |

The external trigger is generated from the TRIGSEL, while the software trigger can be generated by setting the SWTRx bits in the DAC_SWT register.

### 21.3.5. DAC conversion

If the external trigger is enabled by setting the DTENx bit in DAC_CTL0 register, the DAC holding data is transferred to the DAC output data (OUTx_DO) register when the selected trigger event happened. When the external trigger is disabled, the transfer is performed automatically.

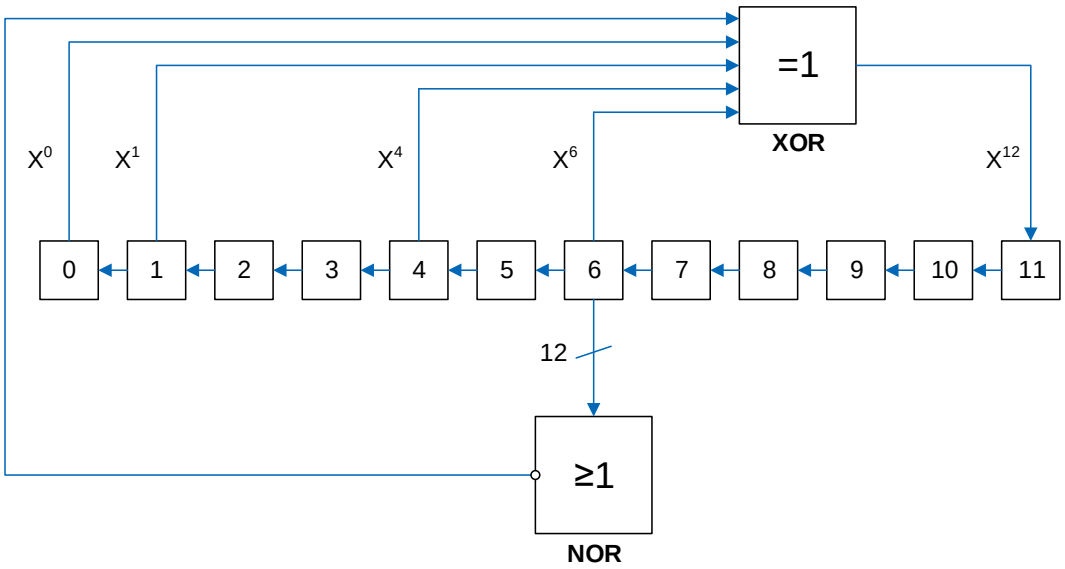
When the DAC holding data (OUTx_DH) is loaded into the OUTx_DO register, after the time $t_{SETTLING}$ which is determined by the analog output load and the power supply voltage, the analog output is valid.

### 21.3.6. DAC noise wave

There are two methods of adding noise wave to the DAC output data: LFSR noise wave mode and Triangle wave mode. The noise wave mode can be selected by the DWMx bits in the DAC_CTL0 register. The amplitude of the noise can be configured by the DAC noise wave bit width (DWBWx) bits in the DAC_CTL0 register.

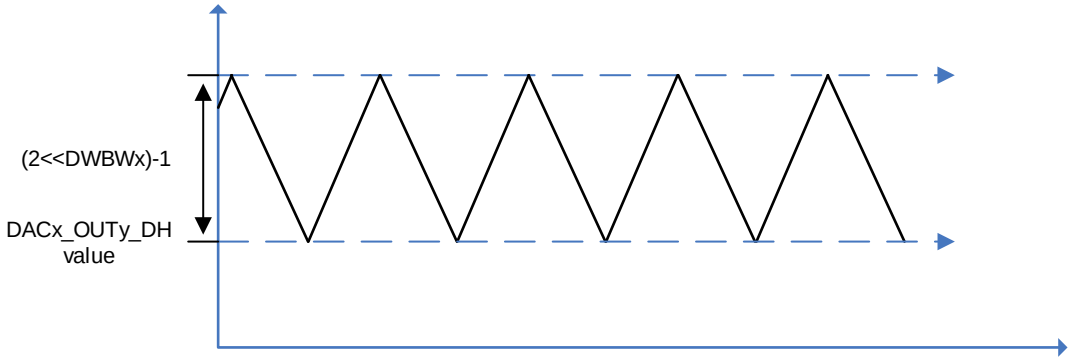
LFSR noise wave mode: there is a Linear Feedback Shift Register (LFSR) in the DAC control logic, it controls the LFSR noise signal which is added to the OUTx_DH value, and then the result is stored into the OUTx_DO register. When the configured DAC noise wave bit width is less than 12, the noise signal equals to the LSB DWBWx bits of the LFSR register, while the MSB bits are masked.

**Figure 21-2. DAC LFSR algorithm**




Triangle noise mode: a triangle signal is added to the OUTx_DH value, and then the result is stored into the OUTx_DO register. The minimum value of the triangle signal is 0, while the maximum value of the triangle signal is (2 << DWBWx) - 1.

**Figure 21-3. DAC triangle noise wave**




### 21.3.7. DAC output voltage

The following equation determines the analog output voltage on the DAC pin.

$$V_{DAC_OUT}=V_{REFP}*OUTx_DO/4096 \quad (4\text{-}1)$$

The digital input is linearly converted to an analog output voltage, its range is 0 to $V_{REFP}$

### 21.3.8. DMA request

When the external trigger is enabled, the DMA request is enabled by setting the DDMAENx bit of the DAC_CTL0 register. A DMA request will be generated when an external hardware trigger (not a software trigger) occurs.

If the second external trigger arrives before confirming the previous request, the new request will not be serviced, and an underrun error event occurs. The DDUDRx bit in the DAC_STAT0 register is set, an interrupt will be generated if the DDUDRIEx bit in the DAC_CTL0 register is set. The DMA request will be stalled until the DDUDRx bit is cleared.

### 21.3.9. DAC concurrent conversion

When the two output channels work at the same time, for maximum bus bandwidth utilization in specific applications, two output channels can be configured in concurrent mode. In concurrent mode, the OUTx_DH and OUTx_DO value will be updated at the same time.

There are three concurrent registers that can be used to load the OUTx_DH value: DACC_R8DH, DACC_R12DH and DACC_L12DH. User just need to access a unique register to realize driving two DAC channels at the same time.

When external trigger is enabled, please ensure both DTENx bits be set, DTSEL0/DTSEL1 bits be same to guarantee the simultaneous trigger.

When DMA is enabled, please ensure any DDMAENx bit in one DAC be set.

The noise mode and noise bit width can be configured either the same or different, depending on the application scenario.

### 21.3.10. DAC output buffer calibration

The output voltage may be offset when DAC use buffer, so it is necessary to compensate output voltage.

The DAC calibration transfer function is:

$$V_{out}=(D/2^{N-1})*G*V_{REFP}+V_{of} \quad (21\text{-}2)$$

Where N is the significant digit of the DAC, D is the digital input of the DAC, and G is the gain, $V_{REFP}$ is the positive reference voltage of the DAC, $V_{of}$ is the offset voltage. The G is 1 and $V_{of}$ is 0 for an ideal DAC.

Calibration will be effective when buffer is enable. During the calibration:

- Buffer disconnect from external pin and on chip peripherals, and enter tri-stated.
- The buffer will be used as a comparator to detect the intermediate code value 0x800,and compare it with $V_{REFP}/2$ through the internal bridge. The CALFx bit in DAC_STAT0 register will be changed to 0 or 1 according to the comparison result.

Two calibration techniques are available:

- Factory calibration(always enable)
  - DAC buffer offset is adjusted at the factory. And the default value of OTVx[4:0] bits in DAC_CALR register is loaded when DAC reset.
- User calibration
  - When the operating conditions are different from the nominal factory calibration conditions, especially when the VDDA, $V_{REFP}$ or temperature change, the user can calibrate at any point in the application process through the software.

The contents of OTVx[4:0] bits can be calculated in a faster way by using successive approximation or dichotomy techniques. The following process is successive approximation method:

- Writing 0 to DENx bit in DAC_CTL0 register to disable DAC output.
- Writing 1 to CALENx bit in DAC_CTL0 register to enable DAC calibration.
- process of calibration algorithm
  - Writing a code (staring by 0x00000b) into OTVx[4:0].
  - Waiting for $T_{cal}$ delay.
  - Checking the flag CALFx in DAC_STAT0 register.
  - If the CALFx set to 1, it proves that the correct calibration value has been found; otherwise, add 1 to the code until the correct calibration value is found.

**Note:** CALENx should write back to 0 after the calibration process, and then wirte DENx to 1 to use DAC at normal mode. It is forbidden to set DENx and CALENx to 1 at the same time.

### 21.3.11. DAC modes

DAC can be set to normal mode or sample and keep mode. The DAC out can be connected to external pin or on chip peripherals.

### Normal mode

When the MODEx[2] bit in the DAC_MDCR register is 0, DAC is in normal mode.

### Sample and keep mode

When the MODEx[2] bit in the DAC_MDCR register is 1, DAC is in sample and keep mode, the DAC core converts the data after triggering the conversion, and then keeps the converted voltage on the capacitor in sample and keep mode. The DAC core is closed between the two samples. Without converting, the DAC output is tri-stated, so the overall power consumption can be reduced. In this mode, LXTAL and IRC32K drive all the corresponding logic and the DAC core and registers, so that DAC can be used in Deep-sleep mode.

The operation of sample and keep mode can be divided into three stages:

#### Sample stage

The sample and keep element is charged to the required voltage. The charging time is determined by the capacitance value. The sampling time is configured by the TSAMPx[9:0] bits in DAC_SKSTRx register. The BWTx bit in DAC_SATA0 is set to 1 when writing the TSAMPx[9:0] bits which indicates the synchronization between AHB clock and IRC32K/LXTAL and the TSAMPx[9:0] bits can be changed by software during DAC out operation.

#### Keep stage

At this stage, the DAC core is closed due to reduced power consumption. The keep time is configured by the TKEEPx[9:0] bits in DAC_SKKTR register. DAC out is tri-stated.

#### Refresh stage

In this stage, DAC core is turn on again to charge the declined voltage to target value.The refresh time is determined by the TREFx[7: 0] bits in DAC_SKRTR register.

When a new OUTx_DH is updated(a trigger when DTENx=1 or update when DTENx = 0), the operation stage will go to sample stage and the DAC core converts the new data to the required voltage. In sample and keep mode, it should take more than 3 cycles of IRC32K between two continuous data update operation for synchronous.

#### Time calculation

The calculation of the time for the three stages above are based on LXTAL/IRC32K clock periods. To configure enough sample and refresh time, refer to the following formula:

**Table 21-4. Formula of sample and refresh time**

| Buffer State | $t_{sample}$ | $t_{refresh}$ | $t_{keep}$ |
|---|---|---|---|
| ON | $t_{wakeup} + R_{BON}*C_{SK}*ln(2^{N+1})$ | $t_{wakeup} + R_{BON}*C_{SK}*ln(2*N_{LSB})$ | $(V_{REFP}/2^{N})*N_{LSB}*C_{SK}/I_{leak}$ |
| OFF | $t_{wakeup} + R_{BOFF}*C_{SK}*ln(2^{N+1})$ | $t_{wakeup} + R_{BOFF}*C_{SK}*ln(2*N_{LSB})$ | $(V_{REFP}/2^{N})*N_{LSB}*C_{SK}/I_{leak}$ |

**Note:**

(1) In the above formula, the $t_{wakeup}$ is wakeup time from off state to the DAC output reaches final value, the charge time is calculated with 1/2 LSB error accuracy to desired output voltage.

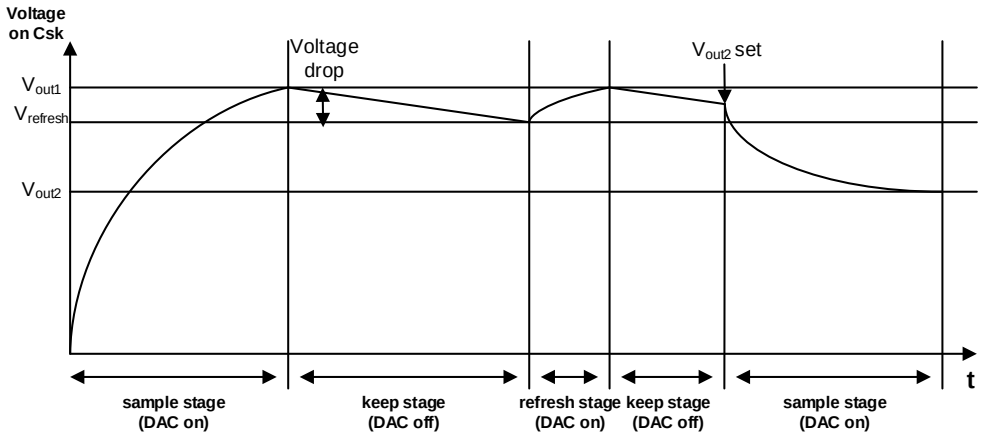
(2) $R_{BON}$/$R_{BOFF}$ is the output impedence when buffer on or off, $C_{SK}$ is the sample and keep capacitor (internal or external). When MODEx[2:0] bits in DAC_MDCR register are 3'b111, the internal capacitor is used to keep the DAC output voltage for on-chip peripherals.

(3) The keep time depends on the tolerance voltage drop during keep stage due to the capacitor discharging with the output leakage current. The number of LSBs $N_{LSB}$ represents the voltage drop, and $I_{leak}$ is the leakage current.

(4) The value of $R_{BON}$, $R_{BOFF}$, $C_{SK}$ and $t_{wakeup}$ please refer to device datasheet.

The sample and keep mode stage diagram is shown as below.

**Figure 21-4. DAC sample and keep**




## 21.3.12. DAC low-power modes

### Sleep mode

In Sleep mode, DAC can work normally, and can be used with DMA.

### Deep-sleep mode

In Deep-sleep mode, if sample and keep mode is enabled before entering Deep-sleep mode, DAC can still hold the static output, otherwise DAC stops working.

### Standby mode

In Standby mode, DAC stops working. When exting from the standby mode, the DAC need to be reinitialized to work again.

## 21.4. DAC register

DAC0 base address: 0x4000 7400

### 21.4.1. DACx control register 0 (DAC_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | CALEN1 | DDUDRIE1 | DDMAEN1 | DWBW1[3:0] | | | | DWM1[1:0] | | Reserved | | DTSEL1[1:0] | | DTEN1 | DEN1 |
| | rw | rw | rw | rw | | | | rw | | | | rw | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | CALEN0 | DDUDRIE0 | DDMAEN0 | DWBW0[3:0] | | | | DWM0[1:0] | | Reserved | | DTSEL0[1:0] | | DTEN0 | DEN0 |
| | rw | rw | rw | rw | | | | rw | | | | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | CALEN1 | DACx_OUT1 calibration enable<br>0: DACx_OUT1 calibration mode disabled<br>1: DACx_OUT1 calibration mode enabled<br>CALEN1 can be written to 1 only when DEN1 is 0 |
| 29 | DDUDRIE1 | DACx_OUT1 DMA underrun interrupt enable<br>0: DACx_OUT1 DMA underrun interrupt disabled<br>1: DACx_OUT1 DMA underrun interrupt enabled |
| 28 | DDMAEN1 | DACx_OUT1 DMA enable<br>0: DACx_OUT1 DMA mode disabled<br>1: DACx_OUT1 DMA mode enabled |
| 27:24 | DWBW1[3:0] | DACx_OUT1 noise wave bit width<br>These bits specify bit width of the noise wave signal of DACx_OUT1. These bits indicate that unmask LFSR bit [n-1, 0] in LFSR noise mode or the amplitude of the triangle is ((2 << (n-1)) - 1) in triangle noise mode, where n is the bit width of wave.<br>0000: The bit width of the wave signal is 1<br>0001: The bit width of the wave signal is 2<br>0010: The bit width of the wave signal is 3<br>0011: The bit width of the wave signal is 4<br>0100: The bit width of the wave signal is 5<br>0101: The bit width of the wave signal is 6<br>0110: The bit width of the wave signal is 7 |

| | | |
|---|---|---|
| | | 0111: The bit width of the wave signal is 8<br>1000: The bit width of the wave signal is 9<br>1001: The bit width of the wave signal is 10<br>1010: The bit width of the wave signal is 11<br>≥1011: The bit width of the wave signal is 12 |
| 23:22 | DWM1[1:0] | DACx_OUT1 noise wave mode<br>These bits specify the mode selection of the noise wave signal of DACx_OUT1 when external trigger of DACx_OUT1 is enabled (DTEN1=1).<br>00: wave disabled<br>01: LFSR noise mode<br>1x: Triangle noise mode |
| 21:20 | Reserved | Must be kept at reset value. |
| 19:18 | DTSEL1[1:0] | DACx_OUT1 trigger selection<br>These bits are only used if bit DTEN = 1 and select the external event used to trigger DAC.<br>00: EXTRIG(external trigger from TRIGSEL).<br>01: Software trigger.<br>All other values: reserved. |
| 17 | DTEN1 | DACx_OUT1 trigger enable<br>0: DACx_OUT1 trigger disabled<br>1: DACx_OUT1 trigger enabled |
| 16 | DEN1 | DACx_OUT1 enable<br>0: DACx_OUT1 disabled<br>1: DACx_OUT1 enabled |
| 15 | Reserved | Must be kept at reset value. |
| 14 | CALEN0 | DACx_OUT0 calibration enable<br>0: DACx_OUT0 calibration mode disabled<br>1: DACx_OUT0 calibration mode enabled<br>CALEN0 can be written to 1 only when DEN0 is 0. |
| 13 | DDUDRIE0 | DACx_OUT0 DMA underrun interrupt enable<br>0: DACx_OUT0 DMA underrun interrupt disabled<br>1: DACx_OUT0 DMA underrun interrupt enabled |
| 12 | DDMAEN0 | DACx_OUT0 DMA enable<br>0: DACx_OUT0 DMA mode disabled<br>1: DACx_OUT0 DMA mode enabled |
| 11:8 | DWBW0[3:0] | DACx_OUT0 noise wave bit width<br>These bits specify bit width of the noise wave signal of DACx_OUT0. These bits indicate that unmask LFSR bit [n-1, 0] in LFSR noise mode or the amplitude of the triangle is ((2<<(n-1))-1) in triangle noise mode, where n is the bit width of wave. |

| | | |
|---|---|---|
| | | 0000: The bit width of the wave signal is 1<br>0001: The bit width of the wave signal is 2<br>0010: The bit width of the wave signal is 3<br>0011: The bit width of the wave signal is 4<br>0100: The bit width of the wave signal is 5<br>0101: The bit width of the wave signal is 6<br>0110: The bit width of the wave signal is 7<br>0111: The bit width of the wave signal is 8<br>1000: The bit width of the wave signal is 9<br>1001: The bit width of the wave signal is 10<br>1010: The bit width of the wave signal is 11<br>≥1011: The bit width of the wave signal is 12 |
| 7:6 | DWM0[1:0] | DACx_OUT0 noise wave mode<br>These bits specify the mode selection of the noise wave signal of DACx_OUT0 when external trigger of DACx_OUT0 is enabled (DTEN0=1).<br>00: Wave disabled<br>01: LFSR noise mode<br>1x: Triangle noise mode |
| 5:4 | Reserved | Must be kept at reset value. |
| 3:2 | DTSEL0[1:0] | DACx_OUT0 trigger selection<br>These bits are only used if bit DTEN = 1 and select the external event used to trigger DAC.<br>00: EXTRIG(external trigger from TRIGSEL).<br>01: Software trigger.<br>All other values: reserved. |
| 1 | DTEN0 | DACx_OUT0 trigger enable<br>0: DACx_OUT0 trigger disabled<br>1: DACx_OUT0 trigger enabled |
| 0 | DEN0 | DACx_OUT0 enable<br>0: DACx_OUT0 disabled<br>1: DACx_OUT0 enabled |

### 21.4.2. DACx software trigger register (DAC_SWT)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | SWTR1 | SWTR0 |
|---|---|---|
| | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | SWTR1 | DACx_OUT1 software trigger, cleared by hardware.<br>0: Software trigger disabled<br>1: Software trigger enabled |
| 0 | SWTR0 | DACx_OUT0 software trigger, cleared by hardware.<br>0: Software trigger disabled<br>1: Software trigger enabled |

### 21.4.3. DACx_OUT0 12-bit right-aligned data holding register (DAC_OUT0_R12DH)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | OUT0_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | OUT0_DH[11:0] | DACx_OUT0 12-bit right-aligned data.<br>These bits specify the data that is to be converted by DACx_OUT0. |

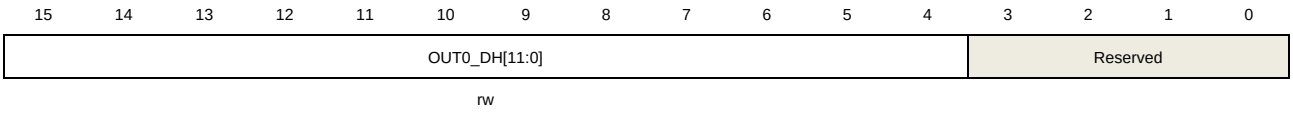
### 21.4.4. DACx_OUT0 12-bit left-aligned data holding register (DAC_OUT0_L12DH)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |



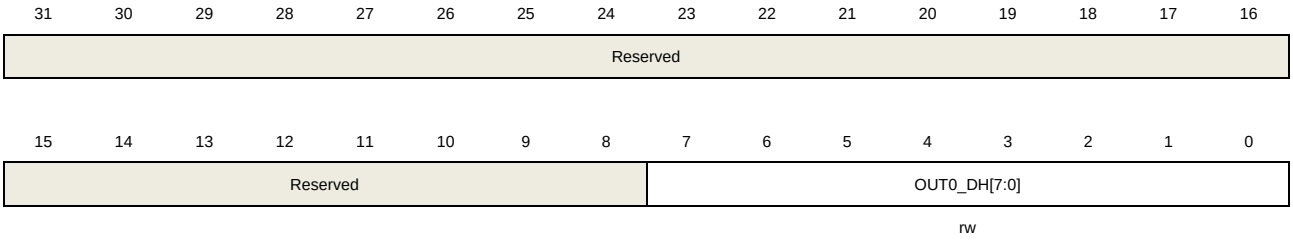

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:4 | OUT0_DH[11:0] | DACx_OUT0 12-bit left-aligned data.<br>These bits specify the data that is to be converted by DACx_OUT0. |
| 3:0 | Reserved | Must be kept at reset value. |

### 21.4.5. DACx_OUT0 8-bit right-aligned data holding register (DAC_OUT0_R8DH)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



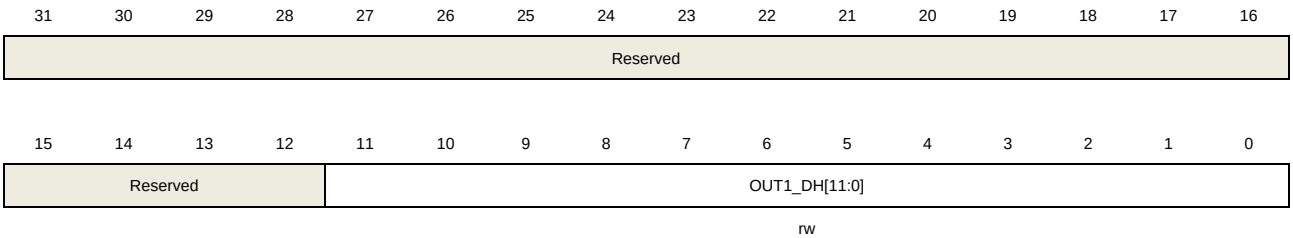

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | OUT0_DH[7:0] | DACx_OUT0 8-bit right-aligned data.<br>These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT0. |

### 21.4.6. DACx_OUT1 12-bit right-aligned data holding register (DAC_OUT1_R12DH)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



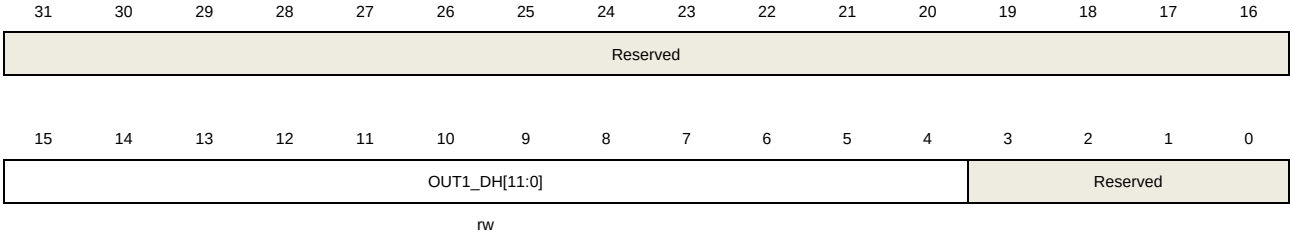
| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | OUT1_DH[11:0] | DACx_OUT1 12-bit right-aligned data.<br>These bits specify the data that is to be converted by DACx_OUT1. |

### 21.4.7. DACx_OUT1 12-bit left-aligned data holding register (DAC_OUT1_L12DH)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



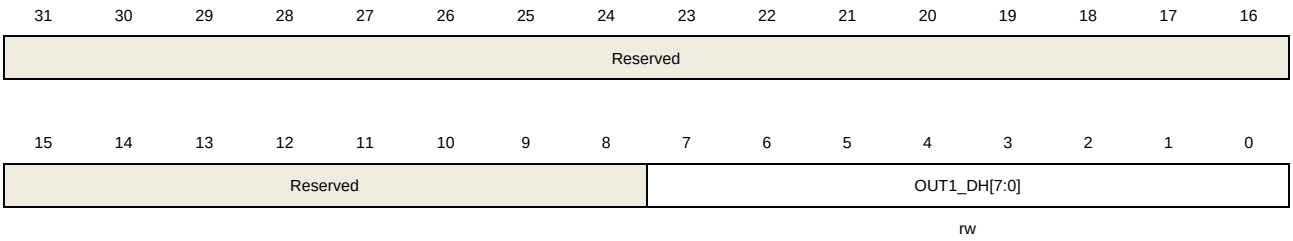

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:4 | OUT1_DH[11:0] | DACx_OUT1 12-bit left-aligned data.<br>These bits specify the data that is to be converted by DACx_OUT1. |
| 3:0 | Reserved | Must be kept at reset value. |

### 21.4.8. DACx_OUT1 8-bit right-aligned data holding register (DAC_OUT1_R8DH)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | OUT1_DH[7:0] | DACx_OUT1 8-bit right-aligned data |

These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT1.

### 21.4.9. DACx concurrent mode 12-bit right-aligned data holding register (DACC_R12DH)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | OUT1_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | OUT0_DH[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | OUT1_DH[11:0] | DACx_OUT1 12-bit right-aligned data<br>These bits specify the data that is to be converted by DACx_OUT1. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | OUT0_DH[11:0] | DACx_OUT0 12-bit right-aligned data<br>These bits specify the data that is to be converted by DACx_OUT0. |

### 21.4.10. DACx concurrent mode 12-bit left-aligned data holding register (DACC_L12DH)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OUT1_DH[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OUT0_DH[11:0] | | | | | | | | | | | | Reserved | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | OUT1_DH[11:0] | DACx_OUT1 12-bit left-aligned data<br>These bits specify the data that is to be converted by DACx_OUT1. |

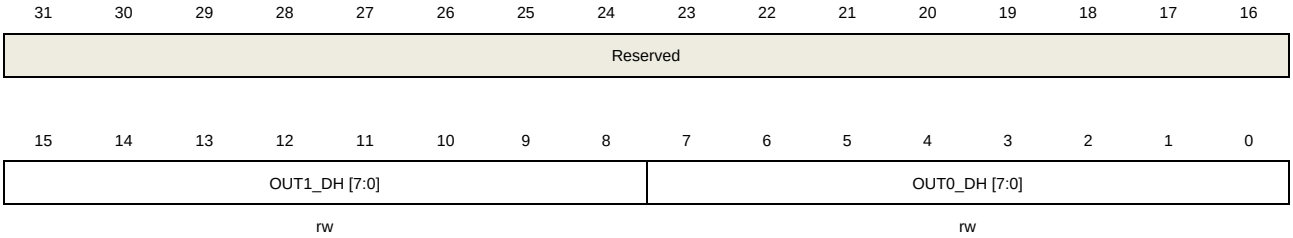
| | | |
|---|---|---|
| 19:16 | Reserved | Must be kept at reset value. |
| 15:4 | OUT0_DH[11:0] | DACx_OUT0 12-bit left-aligned data<br>These bits specify the data that is to be converted by DACx_OUT0. |
| 3:0 | Reserved | Must be kept at reset value. |

### 21.4.11. DACx concurrent mode 8-bit right-aligned data holding register (DACC_R8DH)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



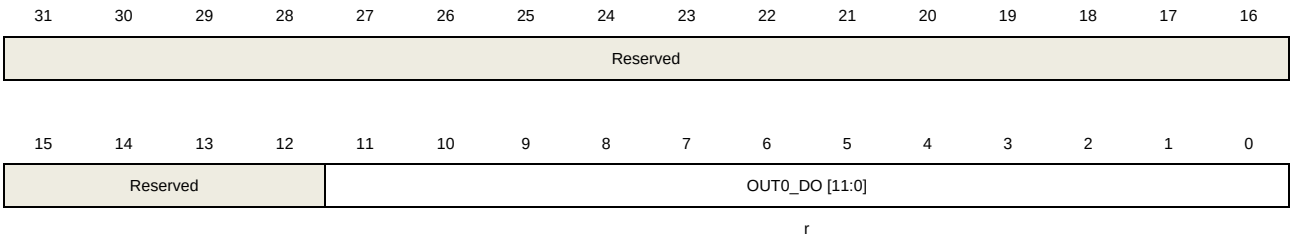

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | OUT1_DH[7:0] | DACx_OUT1 8-bit right-aligned data<br>These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT1. |
| 7:0 | OUT0_DH[7:0] | DACx_OUT0 8-bit right-aligned data<br>These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT0. |

### 21.4.12. DACx_OUT0 data output register (DAC_OUT0_DO)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|

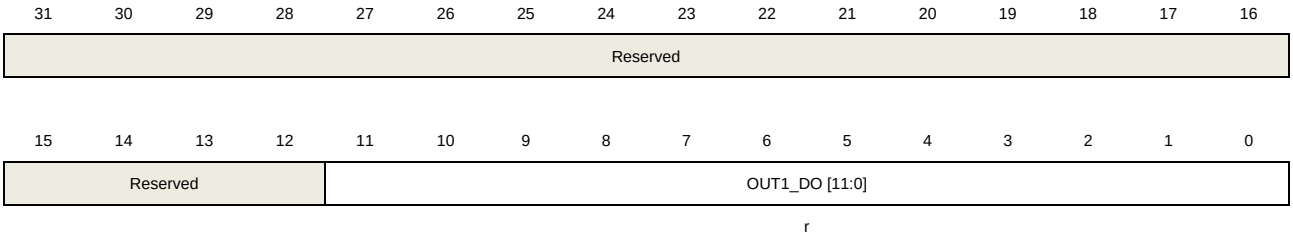
| | | |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | OUT0_DO [11:0] | DACx_OUT0 12-bit output data<br>These bits, which are read only, storage the data that is being converted by DACx_OUT0. |

## 21.4.13. DACx_OUT1 data output register (DAC_OUT1_DO)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



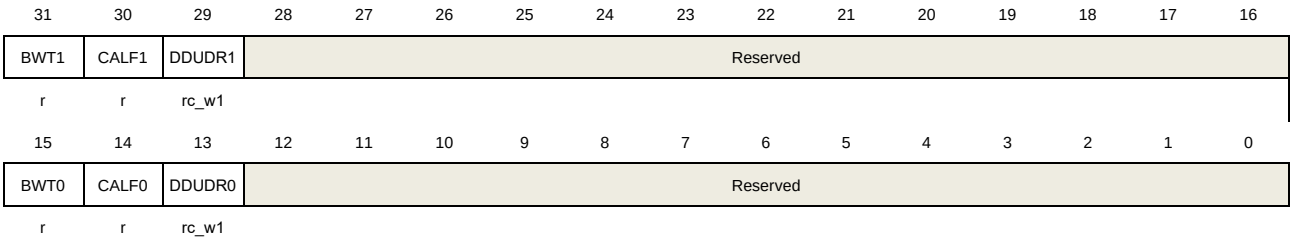

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | OUT1_DO [11:0] | DACx_OUT1 12-bit output data<br>These bits, which are read only, storage the data that is being converted by DACx_OUT1. |

## 21.4.14. DACx status register 0 (DAC_STAT0)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | BWT1 | DACx_OUT1 SKSTR1 writing flag.<br>This bit is set by the system when the sample and keep mode is enabled. When the DACx_SKSTR1 is writing,the bit is set, when the write operation is complete, the bit is cleared by hardware.<br>0: There is no writing operation of DACx_SKSTR1. |

| | | |
|---|---|---|
| | | 1: There is a writing operation of DACx_SKSTR1. |
| 30 | CALF1 | DACx_OUT1 calibration offset flag. This bit is set and cleared by hardware.<br>0: The offset correction value is lower thanl to the calibration value.<br>1: The offset correction value is higher than or equal to the calibration value |
| 29 | DDUDR1 | DACx_OUT1 DMA underrun flag.<br>This bit is set by hardware and cleared by software (by writing it to 1).<br>0: no underrun occurred.<br>1: underrun occurred (Speed of DAC trigger is high than the DMA transfer). |
| 28:16 | Reserved | Must be kept at reset value. |
| 15 | BWT0 | DACx_OUT0 SKSTR0 writing flag.<br>This bit is set by the system when the sample and keep mode is enabled. When the DACx_SKSTR0 is writing,the bit is set, when the write operation is complete, the bit is cleared by hardware.<br>0: There is no writing operation of DACx_SKSTR0.<br>1: There is a writing operation of DACx_SKSTR0. |
| 14 | CALF0 | DACx_OUT0 calibration offset flag. This bit is set and cleared by hardware.<br>0: The offset correction value is lower thanl to the calibration value.<br>1: The offset correction value is higher than or equal to the calibration value |
| 13 | DDUDR0 | DACx_OUT0 DMA underrun flag.<br>This bit is set by hardware and cleared by software (by writing it to 1).<br>0: no underrun occurred.<br>1: underrun occurred (Speed of DAC trigger is high than the DMA transfer). |
| 12:0 | Reserved | Must be kept at reset value. |

## 21.4.15. DACx calibration register (DAC_CALR)

Address offset: 0x38

Reset value: 0x00XX 00XX

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | OTV1 | | | | |
| | | | | | | | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | OTV0 | | | | |
| | | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | OTV1[4:0] | DACx_OUT1 offset calibration value. |

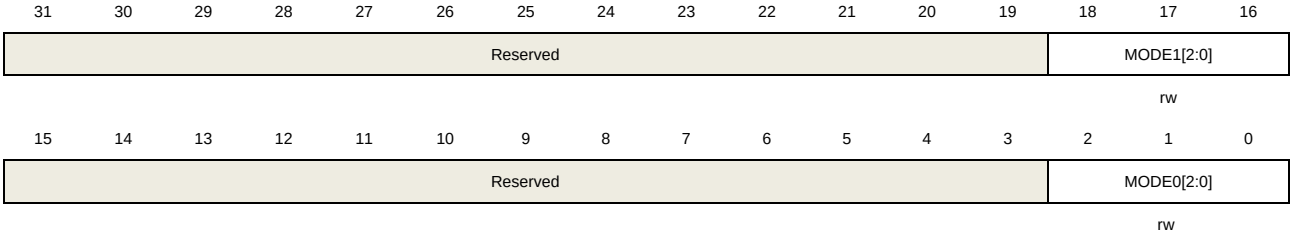
| | | |
|---|---|---|
| 15:5 | Reserved | Must be kept at reset value. |
| 4:0 | OTV0[4:0] | DACx_OUT0 offset calibration value. |

### 21.4.16. DACx mode control register (DAC_MDCR)

Address offset: 0x3C

Reset value: 0x00XX 00XX

This register has to be accessed by word(32-bit).



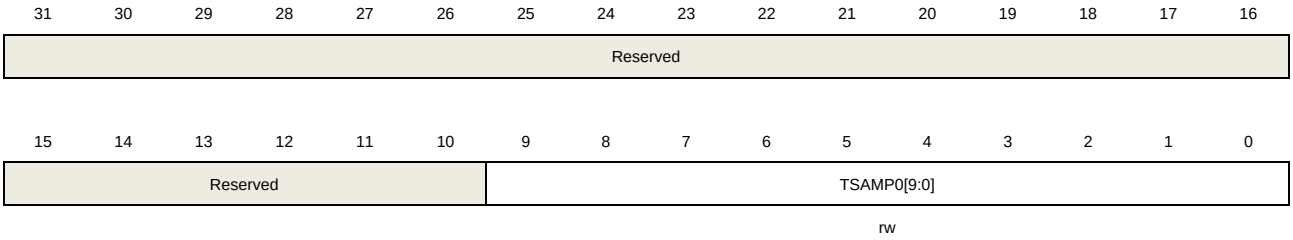

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18:16 | MODE1[2:0] | DACx_OUT1 mode.<br>These bits can be written when bit DEN1=0 and bit CALEN1=0 in the DAC_CTL0 register, the write operation is invalid when DEN1=1 or CALEN1=1.<br>DACx_OUT1 in normal mode<br>000: Buffer is enabled and DACx_OUT1 is connected to external pin<br>001: Buffer is enabled and DACx_OUT1 is connected to on chip peripherals and to external pin.<br>010: Buffer is disabled and DACx_OUT1 is connected to external pin<br>011: Buffer is disabled and DACx_OUT1 is connected to on chip peripherals.<br>DACx_OUT1 in sample and keep mode<br>100: Buffer is enabled and DACx_OUT1 is connected to external pin<br>101: Buffer is enabled and DACx_OUT1 is connected to on chip peripherals and to external pin.<br>110: Buffer is disabled and DACx_OUT1 is connected to on chip peripherals and to external pin.<br>111: Buffer is disabled and DACx_OUT1 is connected to on chip peripherals. |
| 15:3 | Reserved | Must be kept at reset value. |
| 2:0 | MODE0[2:0] | DACx_OUT0 mode.<br>These bits can be written when bit DEN0=0 and bit CALEN0=0 in the DACx_CTL0 register, the write operation is invalid when DEN0=1 or CALEN0=1.<br>DACx_OUT0 in normal mode<br>000: Buffer is enabled and DACx_OUT0 is connected to external pin<br>001: Buffer is enabled and DACx_OUT0 is connected to on chip peripherals and to |

external pin.
010: Buffer is disabled and DACx_OUT0 is connected to external pin
011: Buffer is disabled and DACx_OUT0 is connected to on chip peripherals. DACx_OUT0 in sample and keep mode
100: Buffer is enabled and DACx_OUT0 is connected to external pin
101: Buffer is enabled and DACx_OUT0 is connected to on chip peripherals and to external pin.
110: Buffer is disabled and DACx_OUT0 is connected to on chip peripherals and to external pin.
111: Buffer is disabled and DACx_OUT0 is connected to on chip peripherals.

### 21.4.17. DACx sample and keep sample time register 0 (DAC_SKSTR0)

Address offset: 0x40
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

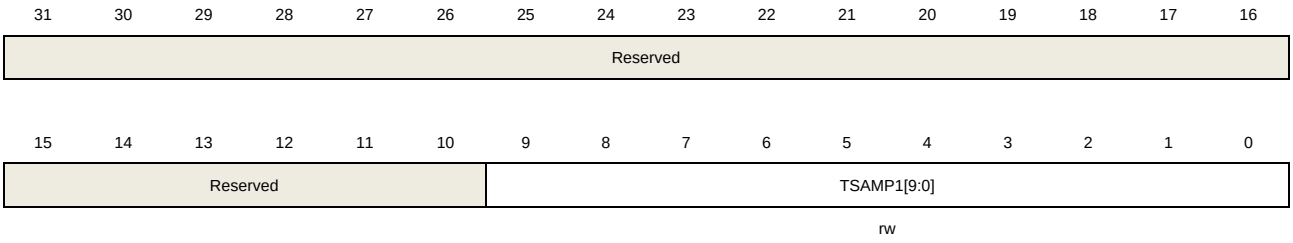



| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9:0 | TSAMP0[9:0] | DACx_OUT0 sample time. |

### 21.4.18. DACx sample and keep sample time register 1 (DAC_SKSTR1)

Address offset: 0x44
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |

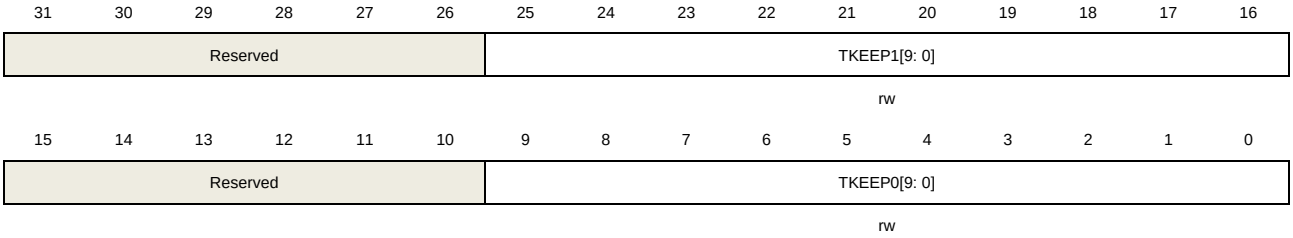
| | | |
|---|---|---|
| 9:0 | TSAMP1[9:0] | DACx_OUT1 sample time. |

## 21.4.19. DACx sample and keep keep time register (DAC_SKKTR)

Address offset: 0x48

Reset value: 0x0001 0001

This register has to be accessed by word(32-bit).



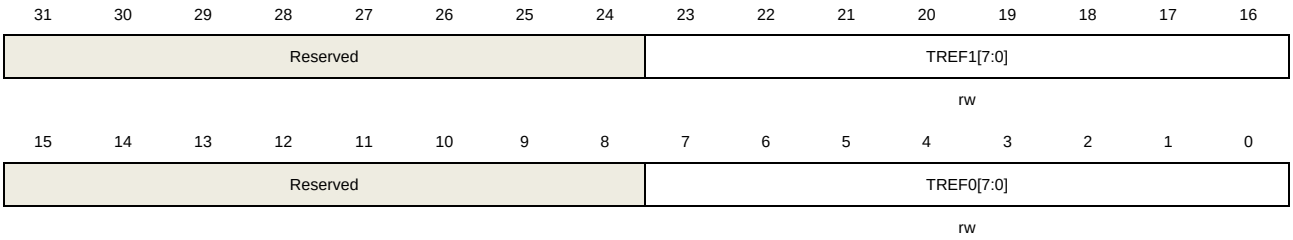

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25:16 | TKEEP1[9: 0] | DACx_OUT1 keep time. |
| 15:10 | Reserved | Must be kept at reset value. |
| 9:0 | TKEEP0[9: 0] | DACx_OUT0 keep time. |

## 21.4.20. DACx sample and keep refresh time register (DAC_SKRTR)

Address offset: 0x4C

Reset value: 0x0001 0001

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | TREF1[7: 0] | DACx_OUT1 refresh time. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | TREF0[7: 0] | DACx_OUT0 refresh time. |

# 22. Watchdog timer (WDGT)

The watchdog timer (WDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. There are two watchdog timer peripherals in the chip: free watchdog timer (FWDGT) and window watchdog timer (WWDGT). They offer a combination of a high safety level, flexibility of use and timing accuracy. Both watchdog timers are offered to resolve malfunctions of software.

The watchdog timer will generate a reset (or an interrupt in window watchdog timer) when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

## 22.1. Free watchdog timer (FWDGT)

### 22.1.1. Overview

The free watchdog timer (FWDGT) has free clock source (IRC32K). Thereupon the FWDGT can operate even if the main clock fails. It's suitable for the situation that requires an independent environment and lower timing accuracy.

The Free watchdog timer generate a reset when the internal down counter reaches 0 or the counter is refreshed when the value of the counter is greater than the window register value. The register write protection function in free watchdog can be enabled to prevent it from changing the configuration unexpectedly.

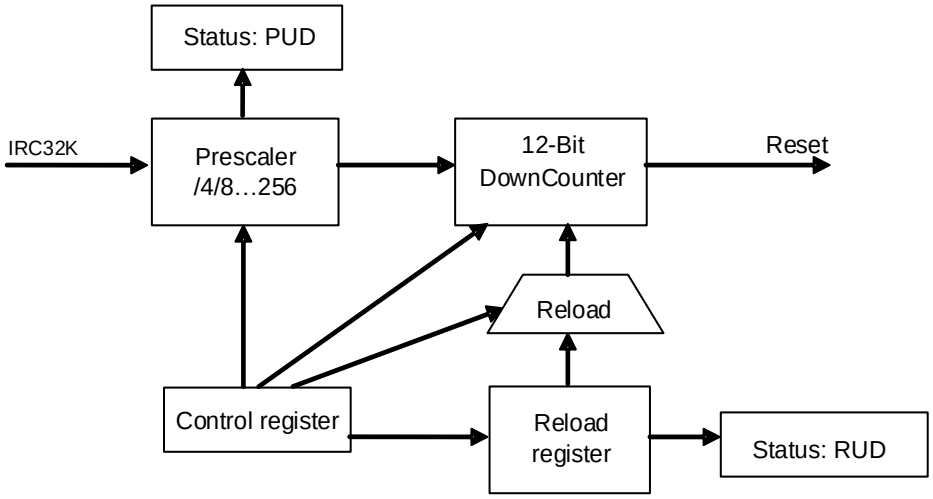
### 22.1.2. Characteristics

- Free-running 12-bit down counter.
- Generate reset in two conditions when FWDGT is enabled:
  - Reset when the counter reached 0.
  - The counter is refreshed when the value of the counter is greater than the window register value.
- Free clock source, FWDGT can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog bit, automatically start the FWDGT or not when power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.
- Configure FWDGSPD_STDBY or FWDGSPD_DPSLP, FWDGT would stop working, or else wake up the device and continue to work, in standby or deep sleep mode.

### 22.1.3. Function overview

The free watchdog consists of an 8-stage prescaler and a 12-bit down counter. ***Figure 22-1. Free watchdog block diagram*** shows the functional block of the free watchdog module.

**Figure 22-1. Free watchdog block diagram**




The free watchdog is enabled by writing the value (0xCCCC) to the control register (FWDGT_CTL), then counter starts counting down. When the counter reaches the value (0x000), there will be a reset.

The counter can be reloaded by writing the value (0xAAAA) to the FWDGT_CTL register at any time. The reload value comes from the FWDGT_RLD register. The software can prevent the watchdog reset by reloading the counter before the counter reaches the value (0x000).

By setting the appropriate window in the FWDGT_WND register, the FWDGT can also work as a window watchdog timer. A reset will occur if the reload operation is performed while the counter is greater than the value stored in the window register (FWDGT_WND). The default value of the FWDGT_WND is 0x0000 0FFF, so if it is not updated, the window option is disabled. A reload operation is performed in order to reset the downcounter to the FWDGT_RLD value and the prescaler counter to generate the next reload, as soon as the window value is changed.

The free watchdog can automatically start at power on when the hardware free watchdog bit in the device option bits is set. To avoid reset, the software should reload the counter before the counter reaches 0x000.

The FWDGT_PSC register, the FWDGT_RLD register and the FWDGT_WND register are write protected. Before writing these registers, the software should write the value (0x5555) to the FWDGT_CTL register. These registers will be protected again by writing any other value to the FWDGT_CTL register. When an update operation of the prescaler register (FWDGT_PSC), window register (FWDGT_WND) or the reload value register (FWDGT_RLD) is ongoing, the status bits in the FWDGT_STAT register are set.

If the FWDGT_HOLD bit in DBG module is cleared, the FWDGT continues to work even the Cortex™-M7 core halted (Debug mode). The FWDGT stops in Debug mode if the FWDGT_HOLD bit is set.

**Table 22-1. Min/max FWDGT timeout period at 32KHz (IRC32K)**

| Prescaler divider | PSC[2:0] bits | Min timeout (ms) RLD[11:0]= 0x000 | Max timeout (ms) RLD[11:0]= 0xFFF |
|---|---|---|---|
| 1/4 | 000 | 0.125 | 512 |
| 1/8 | 001 | 0.25 | 1024 |
| 1/16 | 010 | 0.5 | 2048 |
| 1/32 | 011 | 1.0 | 4096 |
| 1/64 | 100 | 2.0 | 8192 |
| 1/128 | 101 | 4.0 | 16384 |
| 1/256 | 110 or 111 | 8.0 | 32768 |

The FWDGT timeout can be more accurately by calibrating the IRC32K.

**Note:** When after the execution of watchdog reload operation, if the MCU needs enter the deepsleep/standby mode immediately, more than 3 IRC32K clock intervals must be inserted in the middle of reload and deepsleep/standby mode commands by software setting.

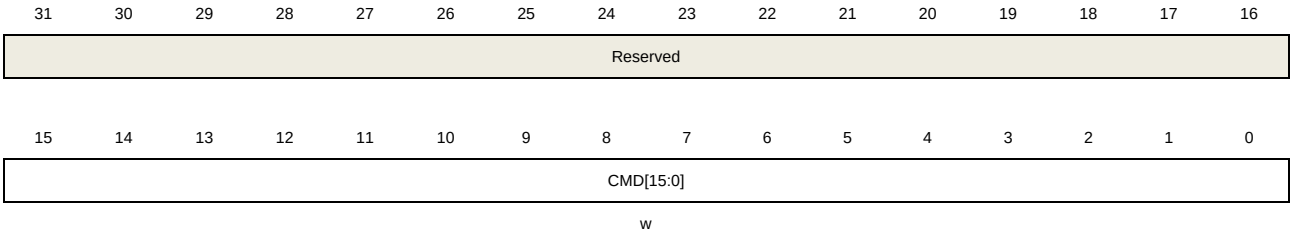
## 22.1.4. Register definition

FWDGT base address: 0x5800 4800

### Control register (FWDGT_CTL)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

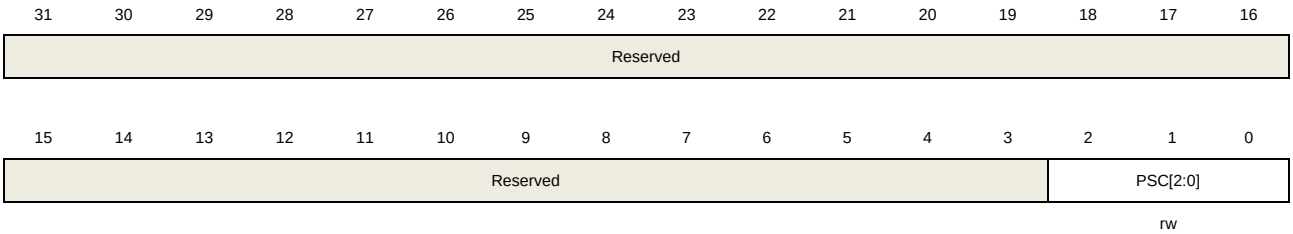



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CMD[15:0] | Write only. Several different functions are realized by writing these bits with different values.<br>0x5555: Disable the FWDGT_PSC, FWDGT_RLD and FWDGT_WND write protection.<br>0xCCCC: Start the free watchdog timer counter. When the counter reduces to 0, the free watchdog generates a reset<br>0xAAAA: Reload the counter |

### Prescaler register (FWDGT_PSC)

Address offset: 0x04
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2:0 | PSC[2:0] | Free watchdog timer prescaler selection. Write 0x5555 in the FWDGT_CTL register before writing these bits. During a write operation to this register, the PUD bit in the FWDGT_STAT register is set and the value read from this register is invalid. |

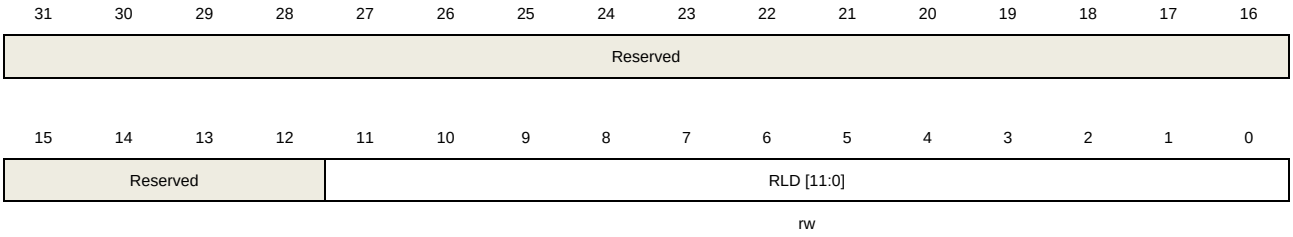
000: 1/4
001: 1/8
010: 1/16
011: 1/32
100: 1/64
101: 1/128
110: 1/256
111: 1/256

If several prescaler values are used by the application, it is mandatory to wait until PUD bit has been reset before changing the prescaler value. If the prescaler value has been updated, it is not necessary to wait until PUD has been reset before continuing code execution (Before entering low-power mode, it is necessary to wait until PUD is reset).

### Reload register (FWDGT_RLD)

Address offset: 0x08
Reset value: 0x0000 0FFF

This register can be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | RLD[11:0] | Free watchdog timer counter reload value. Write 0xAAAA in the FWDGT_CTL register will reload the FWDGT conter with the RLD value.<br>These bits are write-protected. Write 0X5555 to the FWDGT_CTL register before writing these bits. During a write operation to this register, the RUD bit in the FWDGT_STAT register is set and the value read from this register is invalid.<br>If several reload values are used by the application, it is mandatory to wait until RUD bit has been reset before changing the reload value. If the reload value has been updated, it is not necessary to wait until RUD has been reset before continuing code execution (Before entering low-power mode, it is necessary to wait until RUD is reset). |

### Status register (FWDGT_STAT)

Address offset: 0x0C
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | WUD | RUD | PUD |
| | | | | | | | | | | | | | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | WUD | Watchdog counter window value update<br>When a write operation to FWDGT_WND register ongoing, this bit is set and the value read from FWDGT_WND register is invalid. |
| 1 | RUD | Free watchdog timer counter reload value update<br>During a write operation to FWDGT_RLD register, this bit is set and the value read from FWDGT_RLD register is invalid. |
| 0 | PUD | Free watchdog timer prescaler value update<br>During a write operation to FWDGT_PSC register, this bit is set and the value read from FWDGT_PSC register is invalid. |

### Window register (FWDGT_WND)

Address offset: 0x10

Reset value: 0x0000 0FFF

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | WND[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | WND[11:0] | Watchdog counter window value. These bits are used to contain the high limit of the window value to be compared to the downcounter. A reset will occur if the reload operation is performed while the counter is greater than the value stored in this register. The WUD bit in the FWDGT_STAT register must be reset in order to be able to change the reload value.<br>These bits are write protected. Write 0x5555 in the FWDGT_CTL register before writing these bits.<br>If several window values are used by the application, it is mandatory to wait until |

WUD bit has been reset before changing the window value. However, after updating the window value it is not necessary to wait until WUD is reset before continuing code execution except in case of low-power mode entry(Before entering low-power mode, it is necessary to wait until WUD is reset).

# 22.2. Window watchdog timer (WWDGT)

## 22.2.1. Overview

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. After the window watchdog timer starts, the value of down counter reduces progressively. The watchdog timer causes a reset when the counter reached 0x3F (the CNT[6] bit has been cleared). The watchdog timer also causes a reset when the counter is refreshed before the counter reached the window register value. So the software should refresh the counter in a limited window. The window watchdog timer generates an early wakeup status flag when the counter reaches 0x40, interrupt occurs if it is enabled.

The window watchdog timer clock is prescaled from the APB3 clock. The window watchdog timer is suitable for the situation that requires an accurate timing.

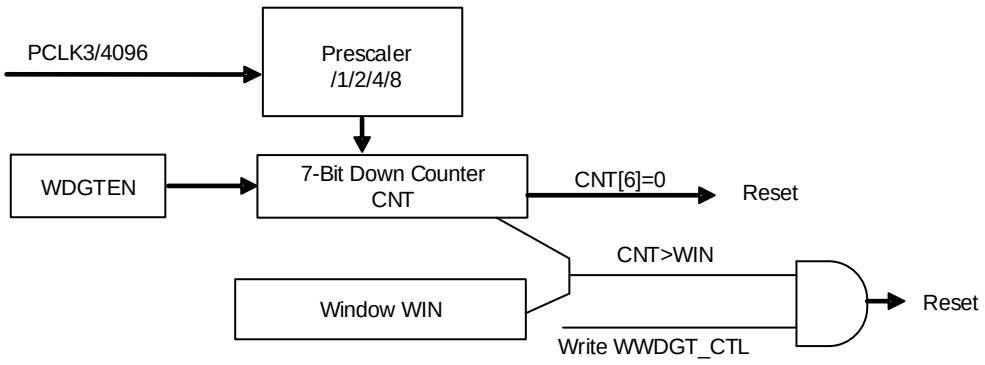
## 22.2.2. Characteristics

- Programmable free-running 7-bit down counter.
- Generate reset in two conditions when WWDGT is enabled:
  - Reset when the counter reached 0x3F.
  - The counter is refreshed when the value of the counter is greater than the window register value.
- Early wakeup interrupt (EWI): if the watchdog is started and the interrupt is enabled, the interrupt occurs when the counter reaches 0x40.
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

## 22.2.3. Function overview

If the window watchdog timer is enabled (set the WDGTEN bit in the WWDGT_CTL), the watchdog timer cause a reset when the counter reaches 0x3F (the CNT[6] bit has been cleared), or the counter is refreshed before the counter reaches the window register value.

**Figure 22-2. Window watchdog timer block diagram**




The watchdog is always disabled after power on reset. The software starts the watchdog by

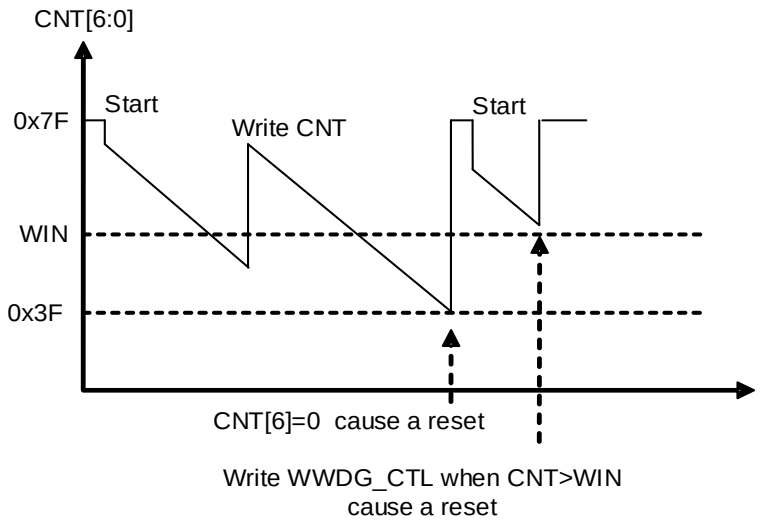
setting the WDGTEN bit in the WWDGT_CTL register. When window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F(it implies that the CNT[6] bit should be set). The CNT[5:0] determine the maximum time interval between two reloading. The count down speed depends on the APB3 clock and the prescaler (PSC[1:0] bits in the WWDGT_CFG register).

The WIN[6:0] bits in the configuration register (WWDGT_CFG) specifies the window value. The software can prevent the reset event by reloading the down counter. The counter value is less than the window value and greater than 0x3F, otherwise the watchdog causes a reset.

The early wakeup interrupt (EWI) is enabled by setting the EWIE bit in the WWDGT_CFG register, and the interrupt will be generated when the counter reaches 0x40. The software can do something such as communication or data logging in the interrupt service routine (ISR) in order to analyse the reason of software malfunctions or save the important data before resetting the device. Moreover the software can reload the counter in ISR to manage a software system check and so on. In this case, the WWDGT will never generate a WWDGT reset but can be used for other things.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDGT_STAT register.

**Figure 22-3. Window watchdog timing diagram**




Calculate the WWDGT timeout by using the formula below.

$$t_{WWDGT}=t_{PCLK3} \times 4096 \times 2^{PSC} \times (CNT[5:0]+1) \quad (ms) \qquad (22\text{-}1)$$

where:

$t_{WWDGT}$: WWDGT timeout

$t_{PCLK3}$: APB3 clock period measured in ms

The ***Table 22-2. Min-max timeout value at 150 MHz (fPCLK3)*** shows the minimum and maximum values of the $t_{WWDGT}$.

**Table 22-2. Min-max timeout value at 150 MHz ($f_{PCLK3}$)**

| Prescaler divider | PSC[1:0] | Min timeout value | Max timeout value |
|---|---|---|---|

| | | CNT[6:0] =0x40 | CNT[6:0]=0x7F |
|---|---|---|---|
| 1/1 | 00 | 27.30 μs | 1.75 ms |
| 1/2 | 01 | 54.61 μs | 3.50 ms |
| 1/4 | 10 | 109.22 μs | 6.99 ms |
| 1/8 | 11 | 218.45 μs | 13.98 ms |

If the WWDGT_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex®-M7 core halted (Debug mode). While the WWDGT_HOLD bit is set, the WWDGT stops in Debug mode.

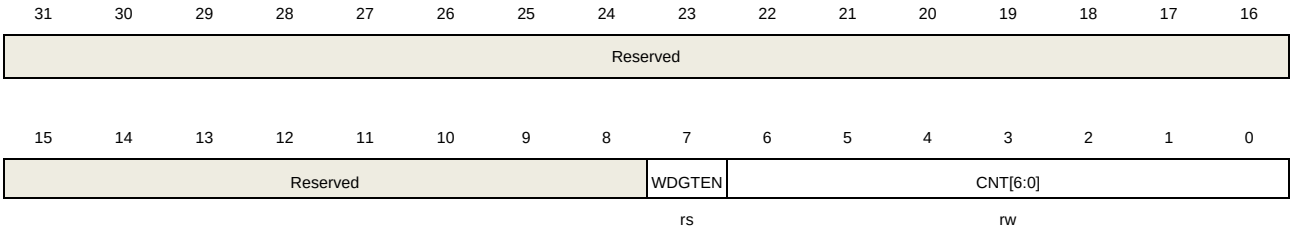
## 22.2.4. Register definition

WWDGT base address: 0x5000 3000

### Control register (WWDGT_CTL)

Address offset: 0x00
Reset value: 0x0000 007F

This register can be accessed by word(32-bit)

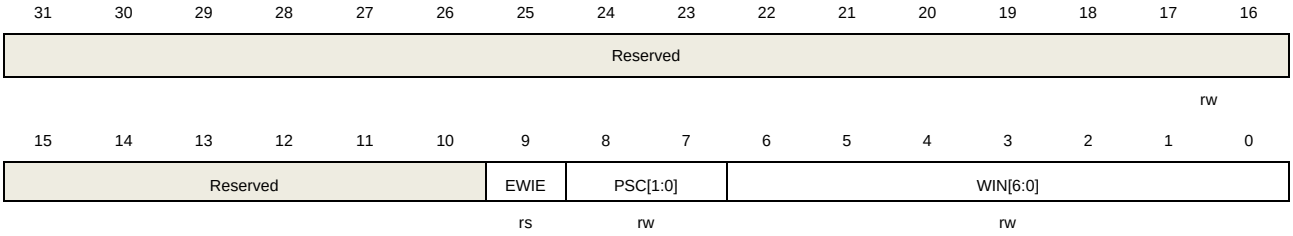



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | WDGTEN | Start the Window watchdog timer. Cleared by a hardware reset. Writing 0 has no effect.<br>0: Window watchdog timer disabled<br>1: Window watchdog timer enabled |
| 6:0 | CNT[6:0] | The value of the watchdog timer counter. A reset occur when the value of this counter decreases from 0x40 to 0x3F. When the value of this counter is greater than the window value, writing this counter also causes a reset. |

### Configuration register (WWDGT_CFG)

Address offset: 0x04
Reset value: 0x0000 007F

This register can be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | EWIE | Early wakeup interrupt enable. If the bit is set, an interrupt occurs when the counter reaches 0x40. It can be cleared by a hardware reset or software clock reset. A write operation of 0 has no effect. |

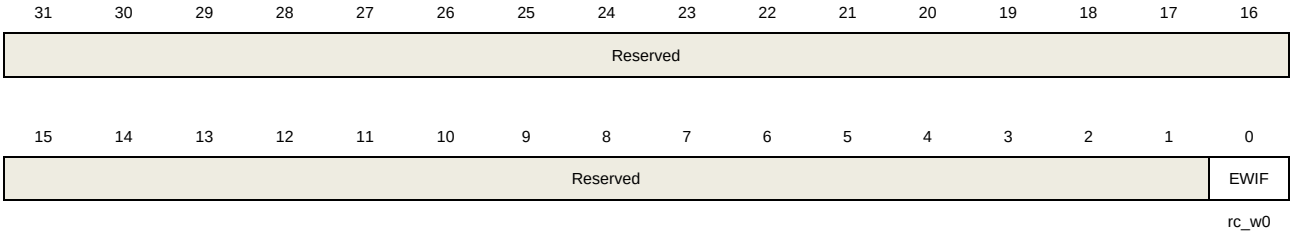
| | | |
|---|---|---|
| 8:7 | PSC[1:0] | Prescaler. The time base of the watchdog counter<br>00: (PCLK3 / 4096) / 1<br>01: (PCLK3 / 4096) / 2<br>10: (PCLK3 / 4096) / 4<br>11: (PCLK3 / 4096) / 8 |
| 6:0 | WIN[6:0] | The Window value. A reset occur if the watchdog counter (CNT bits in WWDGT_CTL) is written when the value of the watchdog counter is greater than the Window value. |

### Status register (WWDGT_STAT)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | EWIF | Early wakeup interrupt flag. When the counter reaches 0x40 or refreshes before it reaches the window value, this bit is set by hardware even the interrupt is not enabled (EWIE in WWDGT_CFG is cleared). This bit is cleared by writing 0. There is no effect when writing 1. |

# 23. Real time clock (RTC)

## 23.1. Overview

The RTC provides a time which includes hour / minute / second / sub-second and a calendar includes year / month / day / week day. The time and calendar are expressed in BCD code except sub-second. Sub-second is expressed in binary code. Hour adjust for daylight saving time. Working in power saving mode and smart wakeup is software configurable. Support improving the calendar accuracy using extern accurate low frequency clock.

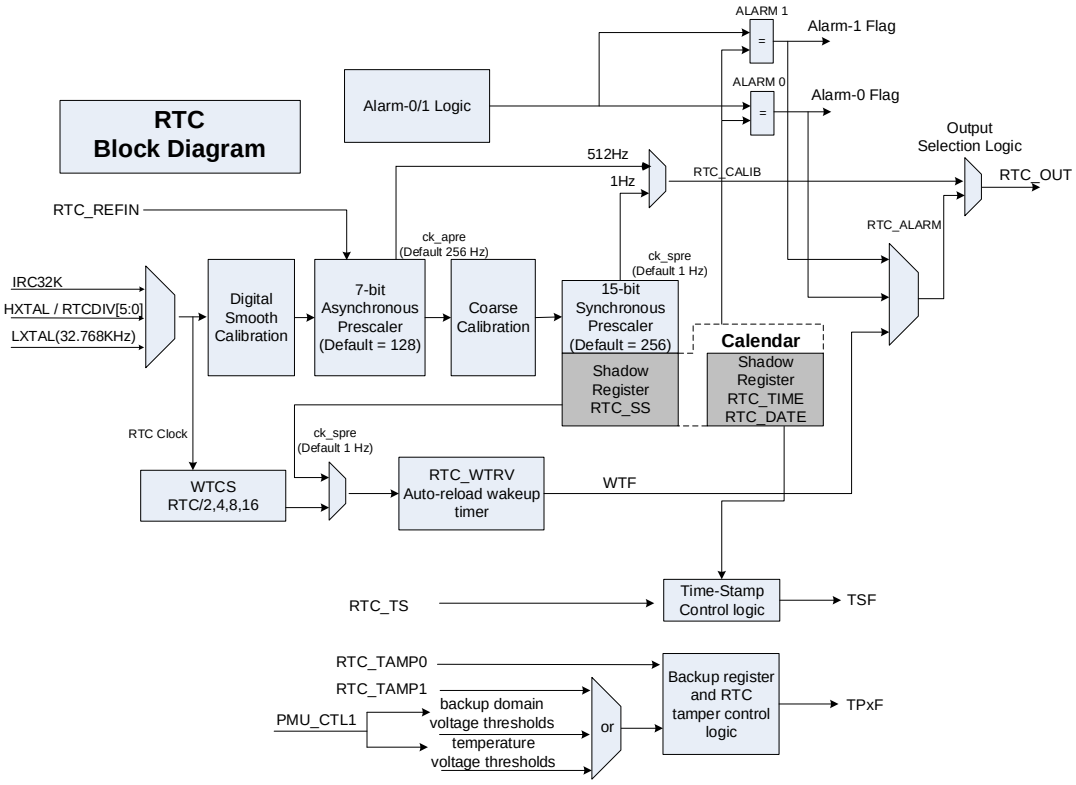
## 23.2. Characteristics

- Support calendar function, which can support year, month, date, day, hours, minutes, seconds and subseconds (date is the day of week and day is the day of month)
- Daylight saving compensation supported, which is realized through software
- External high-accurate low frequency(50Hz or 60Hz) clock used to achieve higher calendar accuracy performed by reference clock detection option function
- Atomic clock adjust (max adjust accuracy is 0.95PPM) for calendar calibration performed by digital calibration function
- Sub-second adjustment by shift function
- Time-stamp function for saving event time
- Two tamper sources can be chosen and tamper type is configurable (RTC_TAMP0 and RTC_TAMP1)
- Programmable calendar and two field maskable alarms
- Maskable interrupt source:
  - Alarm 0 and Alarm 1
  - Time-stamp detection
  - Tamper detection
  - Auto wakeup event
- Thirty-two 32-bit (128 bytes total) universal backup registers which can keep data under power saving mode. Backup register will be reset if tamper event detected

# 23.3. Function overview

## 23.3.1. Block diagram

**Figure 23-1. Block diagram of RTC**




The RTC unit includes:

- Two alarm event / interrupt and support two tamper event / interrupt from I/Os
- When tamper detection event happen, there will generate a timestamp event
- When tamper detection event happen, the backup registers will be erased
- When power switch is switched to the $V_{BAT}$, there will generate a timestamp event
- 32-bit backup registers, which number increased to 32
- Optional RTC output function:
  - 512Hz (default prescale): RTC_OUT(PC13 or PB2)
  - 1Hz(default prescale): RTC_OUT(PC13 or PB2)
  - Alarm event(polarity is configurable): RTC_OUT(PC13 or PB2)
  - Automatic wakeup event(polarity is configurable): RTC_OUT(PC13 or PB2)
- Optional RTC input function:
  - time stamp event detection: RTC_TS(PC13)
  - tamper 0 event detection: RTC_TAMP0(PC13)
  - tamper 1 event detection: RTC_TAMP1(PC1)
  - reference clock input: RTC_REFIN(PB15 or PB13)
  - tamper 1 event detection not only can be generated by I/O, when VBTMEN bit in PMU_CTL1 is set, the backup domain voltage thresholds or temperature voltage thresholds also can generate tamper 1 event detection.

## 23.3.2. Clock source and prescalers

RTC unit has three independent clock sources: LXTAL, IRC32K and HXTAL with divided by

RTCDIV[5:0] (configured in RCU_CFG0 register).

In the RTC unit, there are two prescalers used for implementing the calendar and other functions. One prescaler is a 7-bit asynchronous prescaler and the other is a 15-bit synchronous prescaler. Asynchronous prescaler is mainly used for reducing power consumption. The asynchronous prescaler is recommended to set as high as possible if both prescalers are used.

The frequency formula of two prescalers is shown as below:

$$f_{ck_apre} = \frac{f_{rtcclk}}{FACTOR_A + 1} \quad (23\text{-}1)$$

$$f_{ck_spre} = \frac{f_{ck_apre}}{FACTOR_S + 1} = \frac{f_{rtcclk}}{(FACTOR_A + 1)*(FACTOR_S + 1)} \quad (23\text{-}2)$$

The ck_apre clock is used to driven the RTC_SS down counter which stands for the time left to next second in binary format and when it reaches 0 it will automatically reload FACTOR_S value. The ck_spre clock is used to driven the calendar registers. Each clock will make second plus one.

### 23.3.3. Shadow registers introduction

BPSHAD control bit decides the location when APB bus accesses the RTC calendar register RTC_DATE, RTC_TIME and RTC_SS. By default, the BPSHAD is cleared, and APB bus accesses the shadow calendar registers. Shadow calendar registers is updated with the value of real calendar registers every two RTC clock and at the same time RSYNF bit will be set once. This update mechanism is not performed in Deep-Sleep mode and Standby mode. When exiting these modes, software must clear RSYNF bit and wait it is asserted (the max wait time is 2 RTC clock) before reading calendar register under BPSHAD=0 situation.

**Note**: When reading calendar registers (RTC_SS, RTC_TIME, RTC_DATE) under BPSHAD=0, the frequency of the APB clock ($f_{apb}$) must be at least 7 times the frequency of the RTC clock ($f_{rtcclk}$).

System reset will reset the shadow calendar registers.

### 23.3.4. Configurable and field maskable alarm

RTC alarm function is divided into some fields and each has a maskable bit.

RTC alarm function can be enabled or disabled by ALRMxEN(x=0,1) bit in RTC_CTL. If all the alarm fields value match the corresponding calendar value when ALRMxEN=1(x=0,1), the Alarm flag will be set.

**Note:** FACTOR_S in the RTC_PSC register must be larger than 3 if MSKS bit reset in RTC_ALRMxTD(x=0,1).

If a field is masked, the field is considered as matched in logic. If all the fields have been masked, the Alarm Flag will assert 3 RTC clock later after ALRMxEN(x=0,1) is set.

### 23.3.5. Configurable periodic auto-wakeup counter

In the RTC block, there is a 16-bit down counter designed to generate periodic wakeup flag.

This function is enabled by set the WTEN to 1 and can be running in power saving mode.

Two clock sources can be chose for the down counter:

1) RTC clock divided by 2/4/8/16

   Assume RTC clock comes from LXTAL (32.768 KHz), this can periodically assert wakeup interrupt from 122us to 32s under the resolution down to 61us.

2) Internal clock ck_spre

   Assume ck_spre is 1Hz, this can periodically assert wakeup interrupt from 1s to 36 hours under the resolution down to 1s.

   - WTCS[2:1] = 0b10.This will make period to be 1s to 18 hours
   - WTCS[2:1] = 0b11.This will make period to be 18 to 36 hours

When this function is enabled, the down counter is running. When it reaches 0, the WTF flag is set and the wakeup counter is automatically reloaded with RTC_WUT value.

When WTF asserts, software must then clear it.

If WTIE is set and this counter reaches 0, a wakeup interrupt will make system exit from the power saving mode. System reset has no influence on this function.

WTF is also can be output to RTC_OUT from RTC_ALARM channel.

### 23.3.6. RTC initialization and configuration

#### RTC register write protection

BKPWEN bit in the PMU_CTL register is cleared in default, so writing to RTC registers needs setting BKPWEN bit ahead of time.

After power-on reset, most of RTC registers are write protected. Unlocking this protection is the first step before writing to them.

Following below steps will unlock the write protection:

1. Write ‘0xCA’ into the RTC_WPK register
2. Write ‘0x53’ into the RTC_WPK register

Writing a wrong value to RTC_WPK will make write protection valid again. The state of write protection is not affected by system reset. Following registers are writing protected but others are not:

RTC_TIME, RTC_DATE, RTC_CTL, RTC_STAT, RTC_PSC, RTC_WUT, RTC_ALRM0TD, RTC_ALRM1TD, RTC_SHIFTCTL, RTC_HRFC, RTC_ALRM0SS, RTC_ALRM1SS, RTC_CFG

### Calendar initialization and configuration

The prescaler and calendar value can be programmed by the following steps:

1. Enter initialization mode (by setting INITM=1) and polling INITF bit until INITF=1.
2. Program both the asynchronous and synchronous prescaler factors in RTC_PSC register.
3. Write the initial calendar values into the shadow calendar registers (RTC_TIME and RTC_DATE), and use the CS bit in the RTC_CTL register to configure the time format (12 or 24 hours).
4. Exit the initialization mode (by setting INITM=0).

About 4 RTC clock cycles later, real calendar registers will load from shadow registers and calendar counter restarts.

**Note:** Reading calendar register (BPSHAD=0) after initialization, software should confirm the RSYNF bit to 1.

YCM flag indicates whether the calendar has been initialized by checking the year field of calendar.

### Daylight saving Time

RTC unit supports daylight saving time adjustment through S1H, A1H and DSM bit.

S1H and A1H can subtract or add 1 hour to the calendar when the calendar is running.S1H and A1H operation can be tautologically set and DSM bit can be used to recording this adjust operation. After setting the S1H/A1H, subtract/add 1 hour will perform when next second comes.

### Alarm function operation process

To avoid unexpected alarm assertion and metastable state, alarm function has an operation flow:

1. Disable Alarm (by resetting ALRMxEN(x=0,1) in RTC_CTL)
2. Set the Alarm registers needed(RTC_ALRMxTD/RTC_ALRMxSS(x=0,1))
3. Enable Alarm function (by setting ALRMxEN(x=0,1) in the RTC_CTL)

## 23.3.7. Calendar reading

### Reading calendar registers under BPSHAD=0

When BPSHAD=0, calendar value is read from shadow registers. For the existence of synchronization mechanism, a basic request has to meet: the APB bus clock frequency must be equal to or greater than 7 times the RTC clock frequency.APB bus clock frequency lower than RTC clock frequency is not allowed in any case whatever happens.

When APB bus clock frequency is not equal to or greater than 7 times the RTC clock frequency, the calendar reading flow should be obeyed:

1. reading calendar time register and date register twice

2. if the two values are equal, the value can be seen as the correct value
3. if the two values are not equal, a third reading should performed
4. the third value can be seen as the correct value

RSYNF is asserted once every 2 RTC clock and at this time point, the shadow registers will be updated to current time and date.

To ensure consistency of the 3 values (RTC_SS, RTC_TIME, and RTC_DATE), below consistency mechanism is used in hardware:

1. reading RTC_SS will lock the updating of RTC_TIME and RTC_DATE
2. reading RTC_TIME will lock the updating of RTC_DATE
3. reading RTC_DATE will unlock updating of RTC_TIME and RTC_DATE

If the software wants to read calendar in a short time interval(smaller than 2 RTCCLK periods), RSYNF must be cleared by software after the first calendar read, and then the software must wait until RSYNF is set again before next reading.

In below situations, software should wait RSYNF bit asserted before reading calendar registers (RTC_SS, RTC_TIME, and RTC_DATE):

1. after a system reset
2. after an initialization
3. after shift function

Especially that software must clear RSYNF bit and wait it asserted before reading calendar register after wakeup from power saving mode.

### Reading calendar registers under BPSHAD=1

When BPSHAD=1, RSYNF is cleared and maintains as 0 by hardware so reading calendar registers does not care about RSYNF bit. Current calendar value is read from real-time calendar counter directly. The benefit of this configuration is that software can get the real current time without any delay after wakeup from power saving mode (Deep-sleep /Standby Mode).

Because of no RSYNF bit periodic assertion, the results of the different calendar registers (RTC_SS/RTC_TIME/RTC_DATE) might not be coherent with each other when clock ck_apre edge occurs between two reading calendar registers.

In addition, if current calendar register is changing and at the same time the APB bus reading calendar register is also performing, the value of the calendar register read out might be not correct.

To ensure the correctness and consistency of the calendar value, software must perform reading operation as this: read all calendar registers continuously, if the last two values are the same, the data is coherent and correct.

### 23.3.8. Resetting the RTC

There are two reset sources used in RTC unit: system reset and backup domain reset.

System reset will affect calendar shadow registers and some bits of the RTC_STAT. When system reset is valid, the bits or registers mentioned before are reset to the default value.

Backup domain reset will affect the following registers and system reset will not affect them:

- RTC current real-time calendar registers
- RTC Control register (RTC_CTL)
- RTC Prescaler register (RTC_PSC)
- RTC Wakeup timer register (RTC_WUT)
- RTC High resolution frequency compensation register (RTC_HRFC)
- RTC Shift control register (RTC_SHIFTCTL)
- RTC Time stamp registers (RTC_SSTS/RTC_TTS/RTC_DTS)
- RTC Tamper register (RTC_TAMP)
- RTC Backup registers (RTC_BKPx, RTC_CFG)
- RTC Alarm registers (RTC_ALRMxSS/RTC_ALRMxTD(x=0,1))

The RTC unit will go on running when system reset occurs or enter power saving mode, but if backup domain reset occurs, RTC will stop counting and all registers will reset.

### 23.3.9. RTC shift function

When there is a remote clock with higher degree of precision and RTC 1Hz clock (ck_spre) has an offset (in a fraction of a second) with the remote clock, RTC unit provides a function named shift function to remove this offset and thus make second precision higher.

RTC_SS register indicates the fraction of a second in binary format and is down counting when RTC is running. Therefore by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] or by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] and at the same time set A1S bit can delay or advance the time when next second arrives.

The maximal RTC_SS value depends on the FACTOR_S value in RTC_PSC. The higher FACTOR_S, the higher adjust precision.

Because of the 1Hz clock (ck_spre) is generated by FACTOR_A and FACTOR_S, the higher FACTOR_S means the lower FACTOR_A, then more power consuming.

**Note:** Before using shift function, the software must check the MSB of SSC in RTC_SS (SSC[15]) and confirm it is 0.

After writing RTC_SHIFTCTL register, the SOPF bit in RTC_STAT will be set at once. When shift operation is complete, SOPF bit is cleared by hardware. System reset does not affect SOPF bit.

Shift operation only works correctly when REFEN=0.

Software must not write to RTC_SHIFTCTL if REFEN=1.

### 23.3.10. RTC reference clock detection

RTC reference clock detection is another way to increase the precision of RTC second. To enable this function, you should have an external clock source (50Hz or 60 Hz) which is more precise than LXTAL clock source.

After enabling this function (REFEN=1), each 1Hz clock (ck_spre) edge is compared to the nearest RTC_REFIN clock edge. In most cases, the two clock edges are aligned every time. But when two clock edges are misaligned for the reason of LXTAL poor precision, the RTC reference clock detection function will shift the 1Hz clock edge a little to make next 1Hz clock edge aligned to reference clock edge.

When REFEN=1, a time window is applied at every second update time different detection state will use different window period.

7 ck_apre window is used when detecting the first reference clock edge and 3 ck_apre window is used for the edge aligned operation.

Whatever window used, the asynchronous prescaler counter will be forced to reload when the reference clock is detected in the window. When the two clock (ck_spre and reference clock) edges are aligned, this reload operation has no effect for 1Hz clock. But when the two clock edge are not aligned, this reload operation will shift ck_spre clock edge a bit to make the ck_spre(1Hz) clock edge aligned to the reference clock edge.

When reference detection function is running while the external reference clock is removed (no reference clock edge found in 3 ck_apre window), the calendar updating still can be performed by LXTAL clock only. If the reference clock is recovered later, detection function will use 7 ck_apre window to identify the reference clock and use 3 ck_apre window to adjust the 1Hz clock (ck_spre) edge.

**Note:** Software must configure the FACTOR_A=0x7F and FACTOR_S=0xFF before enabling reference detection function (REFEN=1)

Reference detection function does not work in Standby Mode.

### 23.3.11. RTC smooth digital calibration

RTC smooth calibration function is a way to calibrate the RTC frequency based on RTC clock in a configurable period time.

This calibration is equally executed in a period time and the cycle number of the RTC clock in the period time will be added or subtracted. The resolution of the calibration is about 0.954PPM with the range from -487.1PPM to +488.5PPM.

The calibration period time can be configured to the $2^{20}$/$2^{19}$/$2^{18}$ RTC clock cycles which stands for 32/16/8 seconds if RTC input frequency is 32.768 KHz.

The High resolution frequency compensation register (RTC_HRFC) specifies the number of RTCCLK clock cycles to be calibrated during the period time:

So using CMSK can mask clock cycles from 0 to 511 and thus the RTC frequency can be reduced by up to 487.1PPM.

To increase the RTC frequency the FREQI bit can be set. If FREQI bit is set, there will be 512 additional cycles to be added during period time which means every 211/210/29(32/16/8 seconds) RTC clock insert one cycle.

So using FREQI can increase the RTC frequency by 488.5PPM.

The combined using of CMSK and FREQI can adjust the RTC cycles from -511 to +512 cycles in the period time which means the calibration range is -487.1PPM to +488.5PPM with a resolution of about 0.954PPM.

When calibration function is running, the output frequency of calibration is calculated by the following formula:

$$f_{cal}=f_{rtcclk} \times (1+\frac{FREQI\times 512-CMSK}{2^{N}+CMSK-FREQI\times 512}) \quad (23\text{-}3)$$

**Note:** N=20/19/18 for 32/16/8 seconds window period

### Calibration when FACTOR_A < 3

When asynchronous prescaler value (FACTOR_A) is set to less than 3, software should not set FREQI bit to 1 when using calibration function. FREQI setting will be ignored when FACTOR_A<3.

When the FACTOR_A is less than 3, the FACTOR_S value should be set to a value less than the nominal value. Assuming that RTC clock frequency is nominal 32.768 KHz, the corresponding FACTOR_S should be set as following rule:

FACTOR_A = 2: 2 less than nominal FACTOR_S (8189 with 32.768 KHz)

FACTOR_A = 1: 4 less than nominal FACTOR_S (16379 with 32.768 KHz)

FACTOR_A = 0: 8 less than nominal FACTOR_S (32759 with 32.768 KHz)

When the FACTOR_A is less than 3, CMSK is 0x100, the formula of calibration frequency is as follows:

$$f_{cal}=f_{rtcclk} \times (1+\frac{256-CMSK}{2^{N}+CMSK-256}) \quad (23\text{-}4)$$

**Note:** N=20/19/18 for 32/16/8 seconds window period

### Verifying the RTC calibration

Calibration 1Hz output is provided to assist software to measure and verify the RTC precision.

Up to 2 RTC clock cycles measurement error may occur when measuring the RTC frequency over a limited measurement period. To eliminate this measurement error the measurement period should be the same as the calibration period.

- When the calibration period is 32 seconds(this is default configuration)

Using exactly 32s period to measure the accuracy of the calibration 1Hz output can guarantee

the measure is within 0.477PPM (0.5 RTCCLK cycles over 32s)

- When the calibration period is 16 seconds(by setting CWND16 bit)

In this configuration, CMSK[0] is fixed to 0 by hardware. Using exactly 16s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 0.954PPM (0.5 RTCCLK cycles over 16s)

- When the calibration period is 8 seconds(by setting CWND8 bit)

In this configuration, CMSK[1:0] is fixed to 0 by hardware. Using exactly 8s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 1.907PPM (0.5 RTCCLK cycles over 8s)

#### Re-calibration on-the-fly

When the INITF bit is 0, software can update the value of RTC_HRFC using following steps:

1. Wait the SCPF=0
2. Write the new value into RTC_HRFC register
3. After 3 ck_apre clocks, the new calibration settings take effect

### 23.3.12. Time-stamp function

Time-stamp function is performed on RTC_TS pin and is enabled by control bit TSEN. It is also enabled by control bit ITSEN

When a time-stamp event occurs on RTC_TS pin (TSEN = 1), the calendar value will be saved in time-stamp registers (RTC_DTS/RTC_TTS/RTC_SSTS) and the time-stamp flag (TSF) is set to 1 by hardware. Time-stamp event can generate an interrupt if time-stamp interrupt enable (TSIE) is set.

When an internal time-stamp event detected (ITSEN = 1), the calendar value will be saved in time-stamp registers (RTC_DTS/RTC_TTS/RTC_SSTS), the time-stamp flag (TSF) and internal time-stamp flag (ITSF) is set to 1 by hardware. Time-stamp event can generate an interrupt if internal time-stamp interrupt enable (TSIE) is set.The internal timestamp event is generated by the switch to the $V_{BAT}$ supply

Time-stamp registers only record the calendar at the first time time-stamp event occurs which means that time-stamp registers will not change when TSF=1.

To extend the time-stamp event source, one optional feature is provided: tamper function can also be considered as time-stamp function if TPTS is set.

**Note:** When the time-stamp event occurs, TSF is set 2 ck_apre cycles delay because of synchronization mechanism.

### 23.3.13. Tamper detection

The RTC_TAMPx pin input can be used for tamper event detection under edge detection

mode or level detection mode with configurable filtering setting.

The purposes of the tamper detect configuration are the following:

- The default configuration will erase the RTC backup registers, BKP sram and RTDEC register
- It can wakeup from DeepSleep and Standby modes, and generate an interrupt

### RTC backup registers (RTC_BKPx)

The RTC backup registers are located in the VDD backup domain that remains powered-on by $V_{BAT}$ even if $V_{DD}$ power is switched off. The wake up action from Standby Mode or System Reset does not affect these registers.

These registers are only reset by detected tamper event and backup domain reset.

### Tamper detection function initialization

RTC tamper detection function can be independently enabled on tamper input pin by setting corresponding TPxEN bit. Tamper detection configuration is set before enable TPxEN bit.

The TPxF flag is set after the tamper event occurs on the pin with the following latency:

- When FLT is different from 0x0 (Level detection mode with configurable filtering), there are three ck_apre cycles
- When TPTS is set (Timestamp on tamper event), there are three ck_apre cycles
- When FLT is reset (Edge detection mode on tamper input detection) and TPTS is reset, there is no latency.

When TPxF is set during the latency, new tamper cannot be detected occurring on the same pin.

### Timestamp on tamper event

The TPTS bit can control whether the tamper detection function is used as time-stamp function. If the bit is set to 1, the TSF bit will be set when the tamper event detected as if “enable” the time-stamp function. Whatever the TPTS bit is, the TPxF will assert when tamper event detected.

### Edge detection mode on tamper input detection

When FLT bit is set to 0x0, the tamper detection is set to edge detection mode and TPxEG bit determines the rising edge or falling edge is the detecting edge. When tamper detection is under edge detection mode, the internal pull-up resistors on the tamper detection input pin are deactivated.

Because of detecting the tamper event will reset the backup registers (RTC_BKPx), writing to the backup register should ensure that the tamper event reset and the writing operation will not occur at the same time, a recommend way to avoid this situation is disable the tamper detection before writing to the backup register and re-enable tamper detection after finish

writing.

**Note:** Tamper detection is still running when $V_{DD}$ power is switched off if tamper is enabled.

#### Level detection mode with configurable filtering on tamper input detection

When FLT bit is not reset to 0x0, the tamper detection is set to level detection mode and FLT bit determines the consecutive number of samples (2, 4 or 8) needed for valid level. When DISPU is set to 0x0(this is default), the internal pull-up resistance will pre-charge the tamper input pin before each sampling and thus larger capacitance is allowed to connect to the tamper input pin. The pre-charge duration is configured through PRCH bit. Higher capacitance needs long pre-charge time.

The time interval between each sampling is also configurable. Through adjusting the sampling frequency (FREQ), software can balance between the power consuming and tamper detection latency.

### 23.3.14. Calibration clock output

Calibration clock can be output on the RTC_OUT if COEN bit is set to 1.

When the COS bit is set to 0(this is default) and asynchronous prescaler is set to 0x7F(FACTOR_A), the frequency of RTC_CALIB is $f_{rtcclk}/64$.When the RTCCLK is 32.768KHz, RTC_CALIB output is corresponding to 512Hz.It's recommend to using rising edge of RTC_CALIB output for there may be a light jitter on falling edge.

When the COS bit is set to 1, the RTC_CALIB frequency is:

$$f_{rtc_calib} = \frac{f_{rtcclk}}{(FACTOR_A+1)\times(FACTOR_S+1)} \quad (23\text{-}5)$$

When the RTCCLK is 32.768 KHz, RTC_CALIB output is corresponding to 1Hz if prescaler are default values.

### 23.3.15. Alarm output

When OS control bits are not reset, RTC_ALARM alternate function output is enabled. This function will directly output the content of alarm flag or auto wakeup flag bit in RTC_STAT.

The OPOL bit in RTC_CTL can configure the polarity of the alarm or auto wakeup flag output which means that the RTC_ALARM output is the opposite of the corresponding flag bit or not.

### 23.3.16. RTC pin configuration

RTC_OUT, RTC_TS and RTC_TAMP0 use the same pin (PC13). Function of PC13 is controlled by the RTC and regardless of PC13 GPIO configuration. The RTC functions of PC13 are available in all low-power modes and in VBAT only mode.

The priority of the PC13 output shown in ***Table 23-1 RTC pin configuration***

**Table 23-1 RTC pin configuration and function**

| function configuration and pin function | OS[1:0] (output selection) | COEN (calibration output) | TP0EN (tamper enabled) | TSEN (time stamp enabled) | ALRMOUTTYPE (RTC_ALARM output type) |
|---|---|---|---|---|---|
| Alarm out output open drain | 01 or 10 or 11 | - | - | - | 0 |
| Alarm out output push-pull | 01 or 10 or 11 | - | - | - | 1 |
| Calibration output push-pull | 00 | 1 | - | - | - |
| TAMP0 input floating | 00 | 0 | 1 | 0 | - |
| TIMESTAMP and TAMP0 input floating | 00 | 0 | 1 | 1 | Don't care |
| TIMESTAMP input floating | 00 | 0 | 0 | 1 | Don't care |
| Standard GPIO | 00 | 0 | 0 | 0 | Don't care |

The PC13 can be used for the following purposes:

- RTC_ALARM output: this output can be RTC Alarm 0, RTC Alarm 1 or RTC Wakeup depending on the OS[1:0] bits in the RTC_CTL register
- RTC_CALIB output: this feature is enabled by setting the COEN[23] in the RTC_CTL register
- RTC_TAMP0: tamper event detection
- RTC_TS: time stamp event detection

ALRMOUTTYPE in RTC_CFG is used to select whether the RTC_ALARM is output in push-pull or open-drain mode.

It is possible to output RTC_OUT on PB2 or PC13 pin thanks to OUT2EN bit in RTC_CFG[31]. This output is not available in VBAT / Standby / Shutdown mode.

### 23.3.17. RTC power saving mode management

**Table 23-2 RTC power saving mode management**

| Mode | Active in Mode | Exit Mode |
|---|---|---|
| Sleep | Yes | RTC Interrupts |
| Deep-sleep | Yes: if clock source is LXTAL or IRC32K | RTC Alarm / Tamper Event / Timestamp Event / Wake up |
| Standby | Yes: if clock source is LXTAL or IRC32K | RTC Alarm / Tamper Event / Timestamp Event / Wake up |

### 23.3.18. RTC interrupts

All RTC interrupts are connected to the EXTI controller.

Below steps should be followed if you want to use the RTC alarm/tamper/timestamp/auto wakeup interrupt:

1. Configure and enable the corresponding interrupt line to RTC alarm/tamper/timestamp/auto wakeup event of EXTI and set the rising edge for triggering
2. Configure and enable the RTC alarm/tamper/timestamp/auto wakeup interrupt
3. Configure and enable the RTC alarm/tamper/timestamp/auto wakeup function

**Table 23-3 RTC interrupts control**

| Interrupt | Event flag | Control Bit | Exit Sleep | Exit Deep-sleep And Standby |
|---|---|---|---|---|
| Alarm 0 | ALRM0F | ALRM0IE | Y | Y(1) |
| Alarm 1 | ALRM1F | ALRM1IE | Y | Y(1) |
| Wakeup | WTF | WTIE | Y | Y(1) |
| Timestamp | TSF | TSIE | Y | Y(1) |
| Tamper 0 | TP0F | TPIE | Y | Y(1) |
| Tamper 1 | TP1F | TPIE | Y | Y(1) |

(1) Only active when RTC clock source is LXTAL or IRC32K.

## 23.4. Register definition

RTC base address: 0x5800 4000

### 23.4.1. Time register (RTC_TIME)

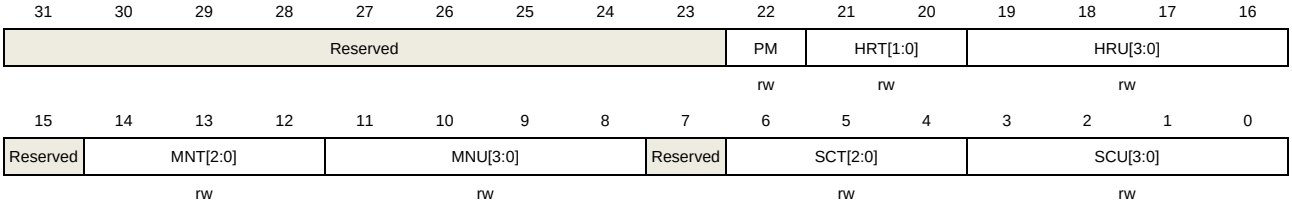
Address offset: 0x00

System reset value: 0x0000 0000 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register is write protected and can only be written in initialization state

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | PM | AM/PM mark<br>0: AM or 24-hour format<br>1: PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | MNT[2:0] | Minute tens in BCD code |
| 11:8 | MNU[3:0] | Minute units in BCD code |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 23.4.2. Date register (RTC_DATE)

Address offset: 0x04

System reset value: 0x0000 2101 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register is write protected and can only be written in initialization state

This register has to be accessed by word (32-bit)

| Reserved | YRT[3:0] | YRU[3:0] |
|---|---|---|
| | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DOW[2:0] | | | MONT | MONU[3:0] | | | | Reserved | | DAYT[1:0] | | DAYU[3:0] | | | |
| rw | | | rw | rw | | | | | | rw | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:20 | YRT | Year tens in BCD code |
| 19:16 | YRU[3:0] | Year units in BCD code |
| 15:13 | DOW[2:0] | Days of the week<br>0x0: Reserved<br>0x1: Monday<br>…<br>0x7: Sunday |
| 12 | MONT | Month tens in BCD code |
| 11:8 | MONU[3:0] | Month units in BCD code |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | DAYT[1:0] | Day tens in BCD code |
| 3:0 | DAYU[3:0] | Day units in BCD code |

### 23.4.3. Control register (RTC_CTL)

Address offset: 0x08

System reset: not affected

Backup domain reset value: 0x0000 0000

This register is writing protected

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | ITSEN | COEN | OS[1:0] | | OPOL | COS | DSM | S1H | A1H |
| | | | | | | | rw | rw | rw | | rw | rw | rw | w | w |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TSIE | WTIE | ALRM1IE | ALRM0IE | TSEN | WTEN | ALRM1EN | ALRM0EN | Reserved | CS | BPSHAD | REFEN | TSEG | WTCS[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | ITSEN | Internal timestamp event enable<br>0: Disable Internal timestamp event<br>1: Enable Internal timestamp event |
| 23 | COEN | Calibration output enable |

| | | |
|---|---|---|
| | | 0: Disable calibration output<br>1: Enable calibration output |
| 22:21 | OS[1:0] | Output selection<br>This bit is used for selecting flag source to output<br>0x0: Disable output RTC_ALARM<br>0x1: Enable alarm0 flag output<br>0x2: Enable alarm1 flag output<br>0x3: Enable wakeup flag output |
| 20 | OPOL | Output polarity<br>This bit is used to invert output RTC_ALARM<br>0: Disable invert output RTC_ALARM<br>1: Enable invert output RTC_ALARM |
| 19 | COS | Calibration output selection<br>Valid only when COEN=1 and prescalers are at default values<br>0: Calibration output is 512 Hz<br>1: Calibration output is 1Hz |
| 18 | DSM | Daylight saving mark<br>This bit is flexible used by software. Often can be used to recording the daylight saving hour adjustment. |
| 17 | S1H | Subtract 1 hour(winter time change)<br>One hour will be subtracted from current time if it is not 0<br>0: No effect<br>1: 1 hour will be subtracted at next second change time. |
| 16 | A1H | Add 1 hour(summer time change)<br>One hour will be added from current time<br>0: No effect<br>1: 1 hour will be added at next second change time |
| 15 | TSIE | Time-stamp interrupt enable<br>0: Disable time-stamp interrupt<br>1: Enable time-stamp interrupt |
| 14 | WTIE | Auto-wakeup timer interrupt enable<br>0: Disable auto-wakeup timer interrupt<br>1: Enable auto-wakeup timer interrupt |
| 13 | ALRM1IE | RTC alarm-1 interrupt enable<br>0: Disable alarm interrupt<br>1: Enable alarm interrupt |
| 12 | ALRM0IE | RTC alarm-0 interrupt enable<br>0: Disable alarm interrupt<br>1: Enable alarm interrupt |

| | | |
|---|---|---|
| 11 | TSEN | Time-stamp function enable<br>0: Disable time-stamp function<br>1: Enable time-stamp function |
| 10 | WTEN | Auto-wakeup timer function enable<br>0: Disable function<br>1: Enable function |
| 9 | ALRM1EN | Alarm-1 function enable<br>0: Disable alarm function<br>1: Enable alarm function |
| 8 | ALRM0EN | Alarm-0 function enable<br>0: Disable alarm function<br>1: Enable alarm function |
| 7 | Reserved | Must be kept at reset value. |
| 6 | CS | Clock System<br>0: 24-hour format<br>1: 12-hour format<br>Note: Can only be written in initialization state |
| 5 | BPSHAD | Shadow registers bypass control<br>0: Reading calendar from shadow registers<br>1: Reading calendar from current real-time calendar<br>Note: If frequency of APB clock is less than seven times the frequency of RTCCLK, this bit must set to 1. |
| 4 | REFEN | Reference clock detection function enable<br>0: Disable reference clock detection function<br>1: Enable reference clock detection function<br>Note: Can only be written in initialization state and FACTOR_S must be 0x00FF |
| 3 | TSEG | Valid event edge of time-stamp<br>0: rising edge is valid event edge for time-stamp event<br>1: falling edge is valid event edge for time-stamp event |
| 2:0 | WTCS[2:0] | Auto-wakeup timer clock selection<br>0x0:RTC Clock divided by 16<br>0x1:RTC Clock divided by 8<br>0x2:RTC Clock divided by 4<br>0x3:RTC Clock divided by 2<br>0x4:0x5: ck_spre (default 1Hz) clock<br>0x6:0x7: ck_spre (default 1Hz) clock and $2^{16}$ is added to wake-up counter. |

### 23.4.4. Status register (RTC_STAT)

Address offset: 0x0C

System reset: Only INITM, INITF and RSYNF bits are set to 0. Others are not affected

Backup domain reset value: 0x0000 0007

This register is writing protected except RTC_STAT[13:8].

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | ITSF | SCPF |
| | | | | | | | | | | | | | | rc_w0 | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TP1F | Reserved | TP0F | TSOVRF | TSF | WTF | ALRM1F | ALRM0F | INITM | INITF | RSYNF | YCM | SOPF | WTWF | ALRM1WF | ALRM0WF |
| rc_w0 | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rw | r | rc_w0 | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | ITSF | Internal timestamp flag<br>Set by hardware when internal time-stamp event is detected.<br>Cleared by software writing 0, and must be cleared together with TSF bit by writing 0 in both bits. |
| 16 | SCPF | Smooth calibration pending flag<br>Set to 1 by hardware when software writes to RTC_HRFC and set to 0 by hardware when smooth calibration window is completed. |
| 15 | TP1F | RTC_TAMP1 detected flag<br>Set to 1 by hardware when tamper detection is found on tamper1 input pin. Software can clear this bit by writing 0 into this bit. |
| 14 | Reserved | Must be kept at reset value. |
| 13 | TP0F | RTC_TAMP0 detected flag<br>Set to 1 by hardware when tamper detection is found on tamper0 input pin. Software can clear this bit by writing 0 into this bit. |
| 12 | TSOVRF | Time-stamp overflow flag<br>This bit is set by hardware when a time-stamp event is detected if TSF bit is set before.<br>Cleared by software writing 0. |
| 11 | TSF | Time-stamp flag<br>Set by hardware when time-stamp event is detected.<br>Cleared by software writing 0. |
| 10 | WTF | Wakeup timer flag<br>Set by hardware when wakeup timer decreased to 0.<br>Cleared by software writing 0.<br>This flag must be cleared at least 1.5 RTC Clock periods before WTF is set to 1 again. |
| 9 | ALRM1F | Alarm-1 occurs flag |

| | | |
|---|---|---|
| | | Set to 1 by hardware when current time/date matches the time/date of alarm 1 setting value.<br>Cleared by software writing 0. |
| 8 | ALRM0F | Alarm-0 occurs flag<br>Set to 1 by hardware when current time/date matches the time/date of alarm 0 setting value.<br>Cleared by software writing 0. |
| 7 | INITM | Enter initialization mode<br>0: Free running mode<br>1: Enter initialization mode for setting calendar time/date and prescaler. Counter will stop under this mode. |
| 6 | INITF | Initialization state flag<br>Set to 1 by hardware and calendar register and prescaler can be programmed in this state.<br>0: Calendar registers and prescaler register cannot be changed<br>1: Calendar registers and prescaler register can be changed |
| 5 | RSYNF | Register synchronization flag<br>Set to 1 by hardware every 2 RTCCLK which will copy current calendar time/date into shadow register. Initialization mode (INITM), shift operation pending flag (SOPF) or bypass mode (BPSHAD) will clear this bit. This bit is also can be cleared by software writing 0.<br>0:Shadow register are not yet synchronized<br>1:Shadow register are synchronized |
| 4 | YCM | Year configuration mark<br>Set by hardware if the year field of calendar date register is not the default value 0.<br>0: Calendar has not been initialized<br>1: Calendar has been initialized |
| 3 | SOPF | Shift function operation pending flag<br>0: No shift operation is pending<br>1: Shift function operation is pending |
| 2 | WTWF | Wakeup timer write enable flag<br>0: Wakeup timer update is not allowed<br>1: Wakeup timer update is allowed |
| 1 | ALRM1WF | Alarm 1 configuration can be write flag<br>Set by hardware if alarm register can be wrote after ALRM1EN bit has reset.<br>0: Alarm registers programming is not allowed<br>1: Alarm registers programming is allowed |
| 0 | ALRM0WF | Alarm 0 configuration can be write flag<br>Set by hardware if alarm register can be wrote after ALRM0EN bit has reset. |

0: Alarm registers programming is not allowed.

1: Alarm registers programming is allowed.

### 23.4.5. Prescaler register (RTC_PSC)

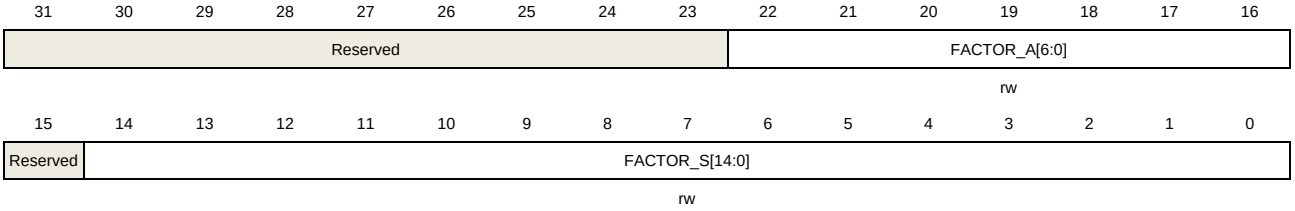
Address offset: 0x10

System reset: not effected

Backup domain reset value: 0x007F 00FF

This register is write protected and can only be written in initialization state

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22:16 | FACTOR_A[6:0] | Asynchronous prescaler factor<br>ck_apre frequency = RTCCLK frequency/(FACTOR_A+1) |
| 15 | Reserved | Must be kept at reset value. |
| 14:0 | FACTOR_S[14:0] | Synchronous prescaler factor<br>ck_spre frequency = ck_apre frequency/(FACTOR_S+1) |

### 23.4.6. Wakeup timer register (RTC_WUT)

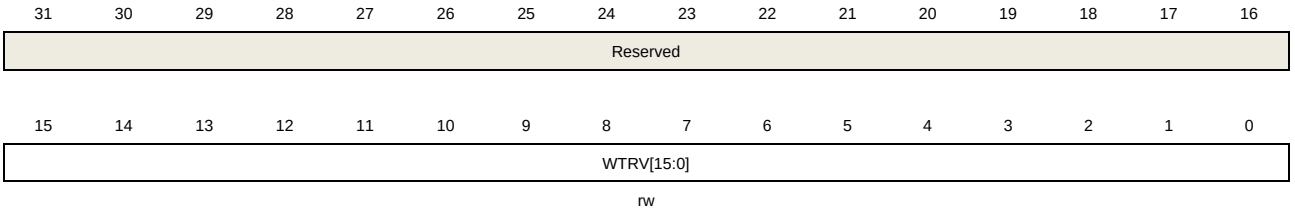
Address offset: 0x14

System reset: not effected

Backup domain reset value: 0x0000 FFFF

This register is writing protected.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | WTRV[15:0] | Auto-wakeup timer reloads value.<br>Every (WTRV[15:0]+1) ck_wut period the WTF bit is set after WTEN=1.The |

ck_wut is selected by WTCS[2:0] bits.

Note: This configure case is forbidden: WTRV=0x0000 with WTCS[2:0]=0b011.

This register can be written only when WTWF=1.

### 23.4.7. Alarm 0 time and date register (RTC_ALRM0TD)

Address offset: 0x1C

System reset: not effect

Backup domain reset value: 0x0000 0000

This register is write protected and can only be written in initialization state

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSKD | DOWS | DAYT[1:0] | | DAYU[3:0] | | | | MSKH | PM | HRT[1:0] | | HRU[3:0] | | | |
| rw | rw | rw | | rw | | | | rw | rw | rw | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSKM | MNT[2:0] | | | MNU[3:0] | | | | MSKS | SCT[2:0] | | | SCU[3:0] | | | |
| rw | rw | | | rw | | | | rw | rw | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MSKD | Alarm date mask bit<br>0: Not mask date/day field<br>1: Mask date/day field |
| 30 | DOWS | Day of the week selected<br>0: DAYU[3:0] indicates the date units<br>1: DAYU[3:0] indicates the week day and DAYT[1:0] has no means. |
| 29:28 | DAYT[1:0] | Date tens in BCD code |
| 27:24 | DAYU[3:0] | Date units or week day in BCD code |
| 23 | MSKH | Alarm hour mask bit<br>0: Not mask hour field<br>1: Mask hour field |
| 22 | PM | AM/PM flag<br>0: AM or 24-hour format<br>1: PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | MSKM | Alarm minutes mask bit<br>0: Not mask minutes field<br>1: Mask minutes field |
| 14:12 | MNT[2:0] | Minutes tens in BCD code |

| | | |
|---|---|---|
| 11:8 | MNU[3:0] | Minutes units in BCD code |
| 7 | MSKS | Alarm second mask bit<br>0: Not mask second field<br>1: Mask second field |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 23.4.8. Alarm 1 time and date register (RTC_ALRM1TD)

Address offset: 0x20
System reset: not effect
Backup domain reset value: 0x0000 0000
This register is write protected and can only be written in initialization state

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSKD | DOWS | DAYT[1:0] | | DAYU[3:0] | | | | MSKH | PM | HRT[1:0] | | HRU[3:0] | | | |
| rw | rw | rw | | rw | | | | rw | rw | rw | | rw | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSKM | MNT[2:0] | | | MNU[3:0] | | | | MSKS | SCT[2:0] | | | SCU[3:0] | | | |
| rw | rw | | | rw | | | | rw | rw | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MSKD | Alarm date mask bit<br>0: Not mask date/day field<br>1: Mask date/day field |
| 30 | DOWS | Day of the week selected<br>0: DAYU[3:0] indicates the date units<br>1: DAYU[3:0] indicates the week day and DAYT[3:0] has no means. |
| 29:28 | DAYT[1:0] | Day tens in BCD code |
| 27:24 | DAYU[3:0] | Day units or week day in BCD code |
| 23 | MSKH | Alarm hour mask bit<br>0: Not mask hour field<br>1: Mask hour field |
| 22 | PM | AM/PM flag<br>0: AM or 24-hour format<br>1: PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | MSKM | Alarm minutes mask bit |

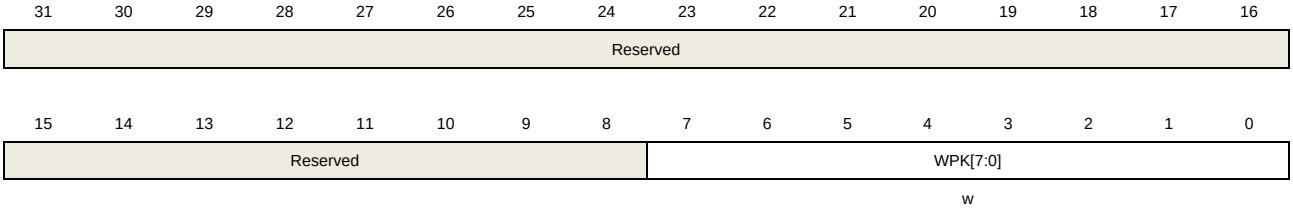
| | | |
|---|---|---|
| | | 0: Not mask minutes field<br>1: Mask minutes field |
| 14:12 | MNT[2:0] | Minutes tens in BCD code |
| 11:8 | MNU[3:0] | Minutes units in BCD code |
| 7 | MSKS | Alarm second mask bit<br>0: Not mask second field<br>1: Mask second field |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 23.4.9. Write protection key register (RTC_WPK)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | WPK[7:0] | Key for write protection |

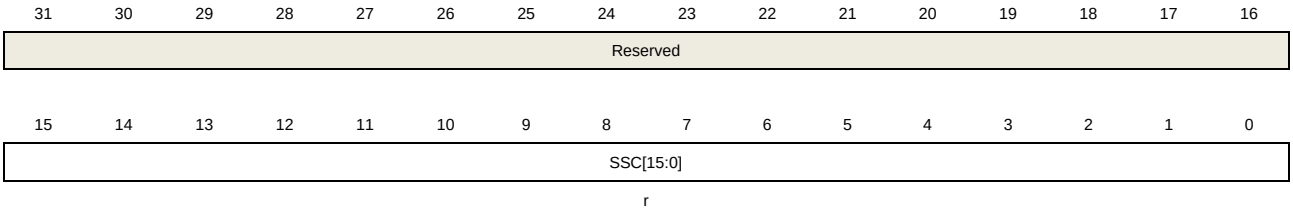
### 23.4.10. Sub second register (RTC_SS)

Address offset: 0x28

System reset value: 0x0000 0000 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | SSC[15:0] | Sub second value |

This value is the counter value of synchronous prescaler. Second fraction value is calculated by the below formula:

Second fraction = ( FACTOR_S - SSC ) / ( FACTOR_S + 1 )

## 23.4.11. Shift function control register (RTC_SHIFTCTL)

Address offset: 0x2C

System reset: not effect

Backup Reset value: 0x0000 0000

This register is writing protected and can only be wrote when SOPF=0

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A1S | Reserved | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | SFS[14:0] | | | | | | | | | | | | | | |
| | w | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | A1S | One second add<br>0: Not add 1 second<br>1: Add 1 second to the clock/calendar.<br>This bit is jointly used with SFS field to add a fraction of a second to the clock. |
| 30:15 | Reserved | Must be kept at reset value. |
| 14:0 | SFS[14:0] | Subtract a fraction of a second<br>The value of this bit will add to the counter of synchronous prescaler.<br>When only using SFS, the clock will delay because the synchronous prescaler is a down counter:<br>Delay (seconds) = SFS / ( FACTOR_S + 1 )<br>When jointly using A1S and SFS, the clock will advance:<br>Advance (seconds) = ( 1 - ( SFS / ( FACTOR_S + 1 ) ) ) |

**Note:** Writing to this register will cause RSYNF bit to be cleared.

## 23.4.12. Time of time stamp register (RTC_TTS)

Address offset: 0x30

Backup domain reset value: 0x0000 0000

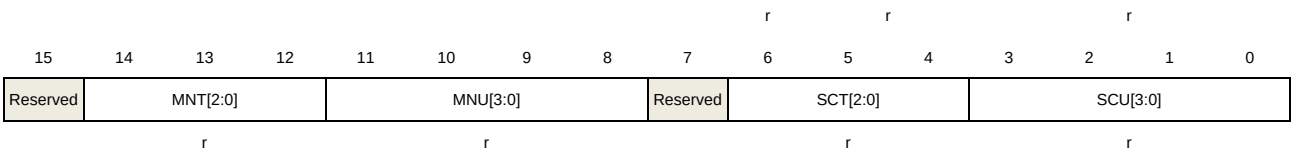
System reset: no effect

This register will record the calendar time when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | PM | HRT[1:0] | | HRU[3:0] | | | |




| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | PM | AM/PM mark<br>0:AM or 24-hour format<br>1:PM |
| 21:20 | HRT[1:0] | Hour tens in BCD code |
| 19:16 | HRU[3:0] | Hour units in BCD code |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | MNT[2:0] | Minute tens in BCD code |
| 11:8 | MNU[3:0] | Minute units in BCD code |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | SCT[2:0] | Second tens in BCD code |
| 3:0 | SCU[3:0] | Second units in BCD code |

### 23.4.13. Date of time stamp register (RTC_DTS)

Address offset: 0x34

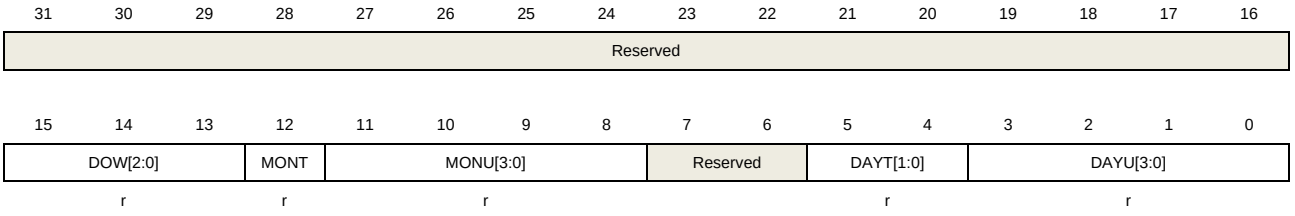
Backup domain reset value: 0x0000 0000

System reset: no effect

This register will record the calendar date when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:13 | DOW[2:0] | Days of the week |
| 12 | MONT | Month tens in BCD code |

| | | |
|---|---|---|
| 11:8 | MONU[3:0] | Month units in BCD code |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | DAYT[1:0] | Day tens in BCD code |
| 3:0 | DAYU[3:0] | Day units in BCD code |

### 23.4.14. Sub second of time stamp register (RTC_SSTS)

Address offset: 0x38

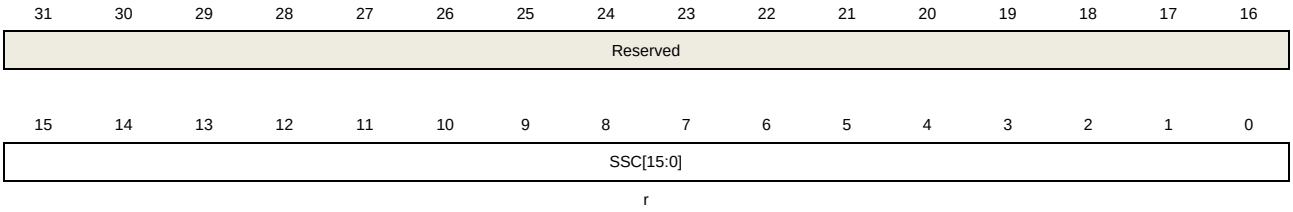
Backup domain reset: 0x0000 0000

System reset: no effect

This register will record the calendar date when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | SSC[15:0] | Sub second value<br>This value is the counter value of synchronous prescaler when TSF is set to 1. |

### 23.4.15. High resolution frequency compensation register (RTC_HRFC)

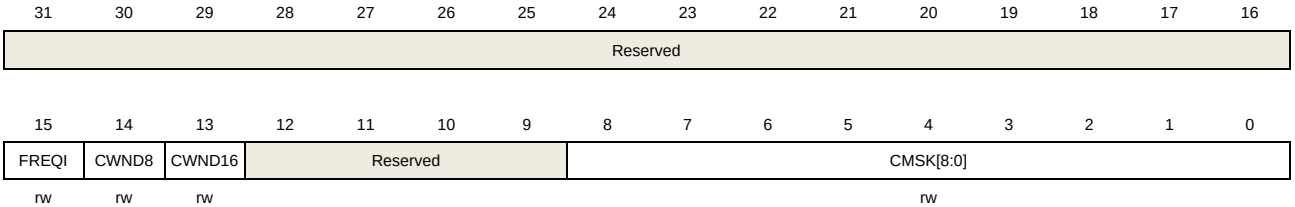
Address offset: 0x3C

Backup domain reset: 0x0000 0000

System Reset: no effect

This register is write protected.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | FREQI | Increase RTC frequency by 488.5PPM<br>0: No effect |

| | | |
|---|---|---|
| | | 1: One RTCCLK pulse is inserted every $2^{11}$ pulses.<br>This bit should be used in conjunction with CMSK bit. If the input clock frequency is 32.768KHz, the number of RTCCLK pulses added during 32s calibration window is (512 * FREQI) - CMSK |
| 14 | CWND8 | Frequency compensation window 8 second selected<br>0: No effect<br>1: Calibration window is 8 second<br>Note: When CWND8=1, CMSK[1:0] are stuck at “00”. |
| 13 | CWND16 | Frequency compensation window 16 second selected<br>0: No effect<br>1: Calibration window is 16 second<br>Note: When CWND16=1, CMSK[0] are stuck at “0”. |
| 12:9 | Reserved | Must be kept at reset value. |
| 8:0 | CMSK[8:0] | Calibration mask number<br>The number of mask pulse out of $2^{20}$ RTCCLK pulse.<br>This feature will decrease the frequency of calendar with a resolution of 0.9537 PPM. |

## 23.4.16. Tamper register (RTC_TAMP)

Address offset: 0x40

Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISPU | PRCH[1:0] | | FLT[1:0] | | FREQ[2:0] | | | TPTS | TP1EG | TP1EN | Reserved | | TPIE | TP0EG | TP0EN |
| rw | rw | | rw | | rw | | | rw | rw | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | DISPU | RTC_TAMPx pull up disable bit<br>0: Enable inner pull-up before sampling for pre-charge RTC_TAMPx pin<br>1: Disable pre-charge duration |
| 14:13 | PRCH[1:0] | Pre-charge duration time of RTC_TAMPx<br>This setting determines the pre-charge time before each sampling.<br>0x0: 1 RTC clock<br>0x1: 2 RTC clock<br>0x2: 4 RTC clock |

| | | |
|---|---|---|
| | | 0x3: 8 RTC clock |
| 12:11 | FLT[1:0] | RTC_TAMPx filter count setting<br>This bit determines the tamper sampling type and the number of consecutive sample.<br>0x0: Detecting tamper event using edge mode. Pre-charge duration is disabled automatically<br>0x1: Detecting tamper event using level mode.2 consecutive valid level samples will make an effective tamper event<br>0x2: Detecting tamper event using level mode.4 consecutive valid level samples will make an effective tamper event<br>0x3: Detecting tamper event using level mode.8 consecutive valid level samples will make an effective tamper event |
| 10:8 | FREQ[2:0] | Sampling frequency of tamper event detection<br>0x0: Sample once every 32768 RTCCLK(1Hz if RTCCLK=32.768KHz)<br>0x1: Sample once every 16384 RTCCLK(2Hz if RTCCLK=32.768KHz)<br>0x2: Sample once every 8192 RTCCLK(4Hz if RTCCLK=32.768KHz)<br>0x3: Sample once every 4096 RTCCLK(8Hz if RTCCLK=32.768KHz)<br>0x4: Sample once every 2048 RTCCLK(16Hz if RTCCLK=32.768KHz)<br>0x5: Sample once every 1024 RTCCLK(32Hz if RTCCLK=32.768KHz)<br>0x6: Sample once every 512 RTCCLK(64Hz if RTCCLK=32.768KHz)<br>0x7: Sample once every 256 RTCCLK(128Hz if RTCCLK=32.768KHz) |
| 7 | TPTS | Make tamper function used for timestamp function<br>0:No effect<br>1:TSF is set when tamper event detected even TSEN=0 |
| 6 | TP1EG | Tamper 1 event trigger edge<br>If tamper detection is in edge mode(FLT =0):<br>0: Rising edge triggers a tamper detection event<br>1: Falling edge triggers a tamper detection event<br>If tamper detection is in level mode(FLT !=0):<br>0: Low level triggers a tamper detection event<br>1: High level triggers a tamper detection event |
| 5 | TP1EN | Tamper 1 detection enable<br>0:Disable tamper 1 detection function<br>1:Enable tamper 1 detection function |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | TPIE | Tamper detection interrupt enable<br>0: Disable tamper interrupt<br>1: Enable tamper interrupt |
| 1 | TP0EG | Tamper 0 event trigger edge<br>If tamper detection is in edge mode(FLT =0): |

0: Rising edge triggers a tamper detection event
1: Falling edge triggers a tamper detection event
If tamper detection is in level mode(FLT !=0):
0: Low level triggers a tamper detection event
1: High level triggers a tamper detection event

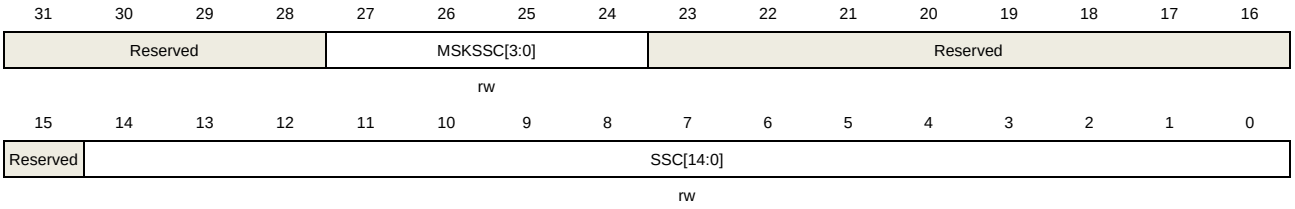
| | | |
|---|---|---|
| 0 | TP0EN | Tamper 0 detection enable<br>0:Disable tamper 0 detection function<br>1:Enable tamper 0 detection function |

**Note:** It's strongly recommended that reset the TPxEN before change the tamper configuration.

## 23.4.17. Alarm 0 sub second register (RTC_ALRM0SS)

Address offset: 0x44
Backup domain reset: 0x0000 0000
System reset: no effect
This register is write protected and can only be wrote when ALRM0EN=0 or INITM=1

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:24 | MSKSSC[3:0] | Mask control bit of SSC<br>0x0: Mask alarm sub second setting. The alarm asserts at every second time point if all the rest alarm fields are matched.<br>0x1: SSC[0] is to be compared and all others are ignored<br>0x2: SSC[1:0] is to be compared and all others are ignored<br>0x3: SSC[2:0] is to be compared and all others are ignored<br>0x4: SSC[3:0] is to be compared and all others are ignored<br>0x5: SSC[4:0] is to be compared and all others are ignored<br>0x6: SSC[5:0] is to be compared and all others are ignored<br>0x7: SSC[6:0] is to be compared and all others are ignored<br>0x8: SSC[7:0] is to be compared and all others are ignored<br>0x9: SSC[8:0] is to be compared and all others are ignored<br>0xA: SSC[9:0] is to be compared and all others are ignored<br>0xB: SSC[10:0] is to be compared and all others are ignored<br>0xC: SSC[11:0] is to be compared and all others are ignored<br>0xD: SSC[12:0] is to be compared and all others are ignored<br>0xE: SSC[13:0] is to be compared and all others are ignored<br>0xF: SSC[14:0] is to be compared and all others are ignored |

| | | |
|---|---|---|
| | | **Note**: The bit 15 of synchronous counter (SSC[15] in RTC_SS) is never compared. |
| 23:15 | Reserved | Must be kept at reset value. |
| 14:0 | SSC[14:0] | Alarm sub second value<br>This value is the alarm sub second value which is to be compared with synchronous prescaler counter SSC. Bit number is controlled by MSKSSC bits. |

## 23.4.18. Alarm 1 sub second register (RTC_ALRM1SS)

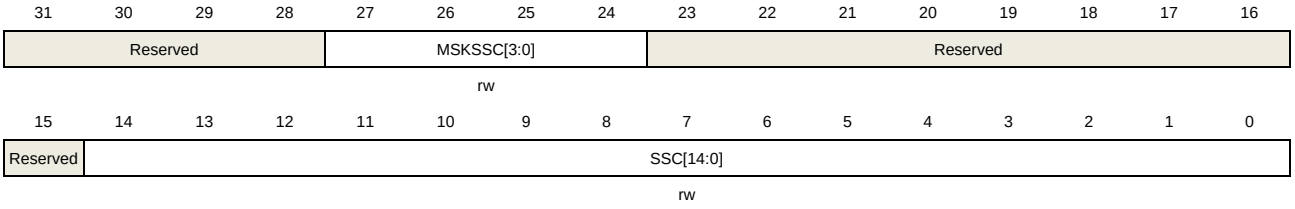
Address offset: 0x48

Backup domain reset: 0x0000 0000

System reset: no effect

This register is write protected and can only be wrote when ALRM1EN=0 or INITM=1

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:24 | MSKSSC[3:0] | Mask control bit of SSC<br>0x0: Mask alarm sub second setting. The alarm asserts at every second time point if all the rest alarm fields are matched.<br>0x1: SSC[0] is to be compared and all others are ignored<br>0x2: SSC[1:0] is to be compared and all others are ignored<br>0x3: SSC[2:0] is to be compared and all others are ignored<br>0x4: SSC[3:0] is to be compared and all others are ignored<br>0x5: SSC[4:0] is to be compared and all others are ignored<br>0x6: SSC[5:0] is to be compared and all others are ignored<br>0x7: SSC[6:0] is to be compared and all others are ignored<br>0x8: SSC[7:0] is to be compared and all others are ignored<br>0x9: SSC[8:0] is to be compared and all others are ignored<br>0xA: SSC[9:0] is to be compared and all others are ignored<br>0xB: SSC[10:0] is to be compared and all others are ignored<br>0xC: SSC[11:0] is to be compared and all others are ignored<br>0xD: SSC[12:0] is to be compared and all others are ignored<br>0xE: SSC[13:0] is to be compared and all others are ignored<br>0xF: SSC[14:0] is to be compared and all others are ignored<br>Note: The bit 15 of synchronous counter (SSC[15] in RTC_SS) is never compared. |

| | | |
|---|---|---|
| 23:15 | Reserved | Must be kept at reset value. |
| 14:0 | SSC[14:0] | Alarm sub second value<br>This value is the alarm sub second value which is to be compared with synchronous prescaler counter SSC. Bit number is controlled by MSKSSC bits. |

## 23.4.19. Configuration register (RTC_CFG)

Address offset: 0x4C

Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | OUT2EN | ALRMOUT TYPE |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | OUT2EN | RTC_OUT pin select<br>0:RTC_OUT is output on PC13<br>1: RTC_OUT is output on PB2 |
| 0 | ALRMOUTTYPE | RTC_ALARM Output Type<br>0: Open-drain output type<br>1: Push-pull output type |

## 23.4.20. Backup registers (RTC_BKPx) (x=0..31)

Address offset: 0x50~0xCC

Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DATA[31:0] | Data<br>These registers can be wrote or read by software. The content remains valid even |

in power saving mode because they can powered-on by VBAT. Tamper detection flag TPxF assertion will reset these registers.

# 24. TIMER (TIMERx)

**Table 24-1. Timers (TIMERx) are divided into five sorts**

| TIMER | | TIMER0/7 | TIMER1/2/3/4/22/23/30/31 | TIMER14/40/41/42/43/44 | TIMER15/16 | TIMER5/6/50/51 |
|---|---|---|---|---|---|---|
| TYPE | | Advanced | General-L0 | General-L3 | General-L4 | Basic |
| Prescaler | | 16-bit | 16-bit | 16-bit | 16-bit | 16-bit |
| Counter | | 16-bit | 16-bit(TIMER2/3/30/31)<br>32-bit(TIMER1/4/22/23) | 16-bit | 16-bit | 32-bit (TIMER5/6)<br>64-bit(TIMER50/51) |
| Count mode | | UP, DOWN, Center-aligned | UP, DOWN, Center-aligned | UP ONLY | UP ONLY | UP ONLY |
| Repetition | | ● | × | ● | ● | × |
| Channel Capture/ Compare | | 8 | 4 | 3 | 2 | 0 |
| Composite PWM mode | | ● | ● | ● | × | × |
| Output match pulse select | | ● | ● | ● | × | × |
| Complementary & Dead-time | | ● | × | ● | ● | × |
| Break function | BREAK0 | ● | × | ● | ● | × |
| | BREAK1 | ● | × | × | × | × |
| Locked break function | | ● | × | ● | ● | × |
| Single Pulse | | ● | ● | ● | ● | ● |
| Delayable single pulse | | ● | ● | ● | × | × |
| Quadrature decoder | | ● | ● | × | × | × |
| Non-quadrature Decoder | | ● | ● | × | × | × |
| Master-slave management | | ● | ● | ● | × | × |

| TIMER | TIMER0/7 | TIMER1/2/3/4/22/23/30/31 | TIMER14/40/41/42/43/44 | TIMER15/16 | TIMER5/6/50/51 |
|---|---|---|---|---|---|
| Inter Connection | ● (1) | ● (2) | ● (3) | × | TRGO TO DAC |
| DMA | ● | ● | ● | ● | ● (4) |
| Debug Mode | ● | ● | ● | ● | ● |

| | TIMERx | ITI0 | ITI1 | ITI2 | ITI3 | ITI4 | ITI5 | ITI6 | ITI7 | ITI8 | ITI9 | ITI10 | ITI11 | ITI12 | ITI13 | ITI14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (1) | TIMER0 | TIMER14_TRGO0 | TIMER1_TRGO0 | TIMER2_TRGO0 | TIMER3_TRGO0 | - | - | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| | TIMER7 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER3_TRGO0 | TIMER4_TRGO0 | - | - | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| (2) | TIMER1 | TIMER0_TRGO0 | TIMER7_TRGO0 | TIMER2_TRGO0 | TIMER3_TRGO0 | ETH_PPS | USB0OTG_HS_SOF | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| | TIMER2 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER14_TRGO0 | TIMER3_TRGO0 | ETH_PPS | - | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| | TIMER3 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER2_TRGO0 | TIMER7_TRGO0 | - | - | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| | TIMER4 | TIMER0_TRGO0 | TIMER7_TRGO0 | TIMER2_TRGO0 | TIMER3_TRGO0 | - | - | | USB1OTG_HS_SOF | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| | TIMER22 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER2_TRGO0 | TIMER3_TRGO0 | TIMER4_TRGO0 | TIMER7_TRGO0 | - | - | - | TIMER14_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | TIMER23_TRGO0 | |
| | TIMER23 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER2_TRGO0 | TIMER3_TRGO0 | TIMER4_TRGO0 | TIMER7_TRGO0 | - | - | - | TIMER14_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | TIMER22_TRGO0 | - | from TRIGSEL |
| | TIMER30 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER14_TRGO0 | TIMER3_TRGO0 | ETH1_PPS | USB1OTG_HS_SOF | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| | TIMER31 | TIMER0_TRGO0 | TIMER1_TRGO0 | TIMER14_TRGO0 | TIMER3_TRGO0 | ETH1_PPS | - | - | - | - | - | - | - | TIMER22_TRGO0 | TIMER23_TRGO0 | |
| (3) | TIMER14 | TIMER0_TRGO0 | TIMER2_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | - | - | - | - | - | - | - | - | - | |
| | TIMER40 | TIMER0_TRGO0 | TIMER2_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | - | - | - | - | - | - | - | - | - | |
| | TIMER41 | TIMER0_TRGO0 | TIMER2_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | - | - | - | - | - | - | - | - | - | |
| | TIMER42 | TIMER0_TRGO0 | TIMER2_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | - | - | - | - | - | - | - | - | - | |
| | TIMER43 | TIMER0_TRGO0 | TIMER2_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | - | - | - | - | - | - | - | - | - | |
| | TIMER44 | TIMER0_TRGO0 | TIMER2_TRGO0 | TIMER15_CH0 | TIMER16_CH0 | - | - | - | - | - | - | - | - | - | - | |

| | TIMERx | ITI0 | ITI1 | ITI2 | ITI3 | ITI4 | ITI5 | ITI6 | ITI7 | ITI8 | ITI9 | ITI10 | ITI11 | ITI12 | ITI13 | ITI14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (4) | Only update events will generate a DMA request. TIMER5/6/50/51 do not have DMAS bit (DMA request source selection). | | | | | | | | | | | | | | | |

# 24.1. Advanced timer (TIMERx, x=0, 7)

## 24.1.1. Overview

The advanced timer module (TIMER0/7) is an eight-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The advanced timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the advanced timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time insertion module which is suitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counter value increasing in unison.

## 24.1.2. Characteristics

- Total channel num: 8.
- Counter width: 16 bits.
- Selectable clock source: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: up counting, down counting and center-aligned counting.
- Quadrature decoder: used for motion tracking and determination of both rotation direction and position.
- Hall sensor function: used for 3-phase motor control.
- Programmable prescaler: 16 bits. The factor can be changed ongoing.
- Each channel is independent and user-configurable: input capture mode, output compare mode, programmable PWM mode, single pulse mode and trigger out.
- Programmable dead time insertion and separated dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input function: BREAK0 and BREAK1.
- Interrupt output or DMA request: update event, trigger event, compare/capture event and break input.
- Daisy chaining of timer module allows a single timer to start multiple timers.
- Timer synchronization allows the selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 24.1.3. Block diagram

*Figure 24-1. Advanced timer block diagram* provides details of the internal configuration of the advanced timer, and *Table 24-2. Advanced timer channel description* introduces the input and output of the channels.

**Figure 24-1. Advanced timer block diagram**

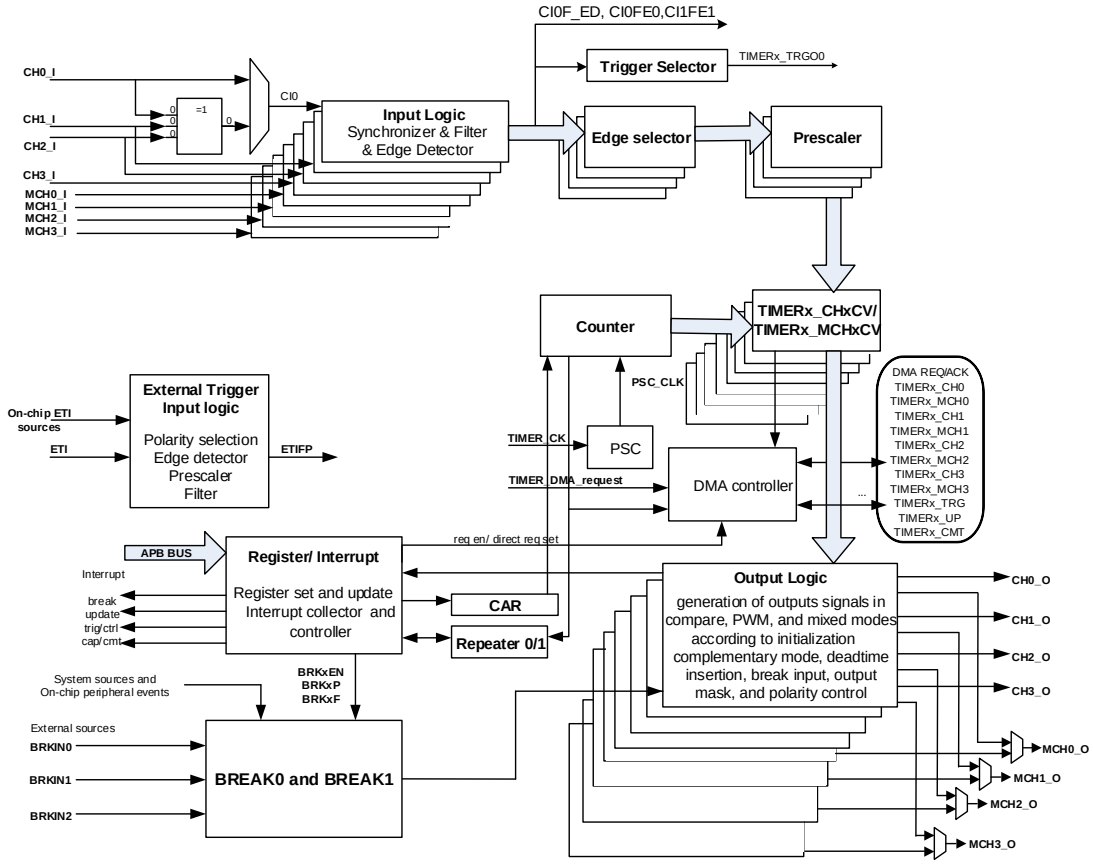



**Table 24-2. Advanced timer channel description**

| Channel name (x=0..3) | MCHxMSEL[1:0]=00 independent mode | MCHxMSEL[1:0]=11 complementary mode |
|---|---|---|
| CHx (Channel x) | CHx and MCHx can independently input capture and compare output | only the CHx is valid for input, and the outputs of MCHx and CHx are complementary |
| MCHx (Multi mode channel x) | | |

### 24.1.4. Function overview

#### Clock selection

The clock source of the advanced timer can be either the CK_TIMER or an alternate clock source controlled by TSCFGy[4:0] (y=0...9,15) in SYSCFG_TIMERxCFG(x=0,7) registers.

- TSCFGy[4:0] (y=0..9,15) = 5'b00000 in SYSCFG_TIMERxCFG(x=0,7) registers. Internal

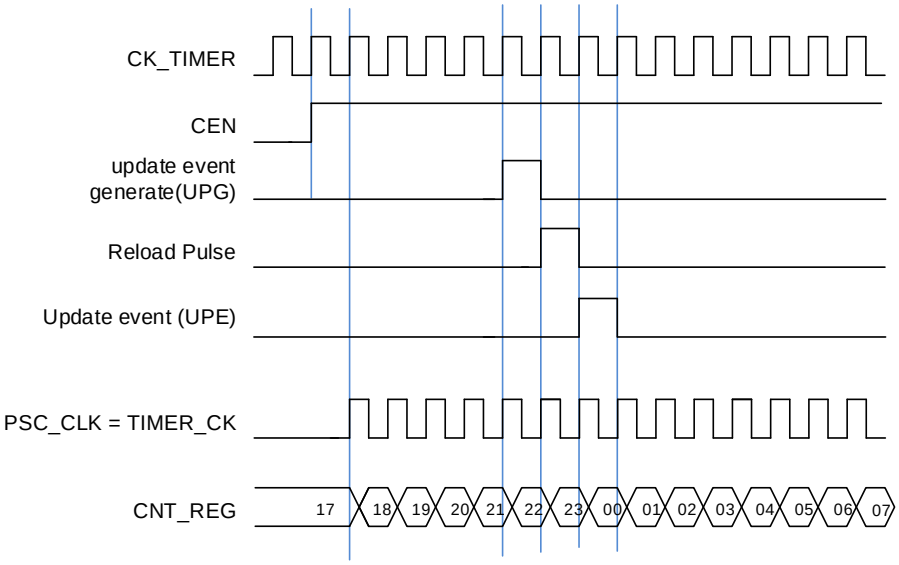
clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when TSCFGy[4:0] (y=0..9,15) = 5'b00000 in SYSCFG_TIMERxCFG(x=0,7) registers. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK which drives counter's prescaler to count is equal to CK_TIMER which is from RCU module.

If TSCFGy[4:0] (y=0..2,6,8,9) in SYSCFG_TIMERxCFG(x=0,7) registers are setting to a nonzero value, the prescaler is clocked by other clock sources selected in the TSCFGy[4:0] (y=0..2,6,8,9) bit-filed, more details will be introduced later. When the TSCFGy[4:0] (y=3,4,5,7) are setting to a nonzero value, the internal clock TIMER_CK is the counter prescaler driving clock source.

**Figure 24-2. Normal mode, internal clock divided by 1**




- TSCFG6[4:0] are setting to a nonzero value (external clock mode 0). External input pin is selected as timer clock source.

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CHn/ TIMERx_MCHn(n=0...3). This mode can be selected by setting TSCFG6[4:0] to 0x5~0x7 and 0x9~0xE.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3/12/13/14. This mode can be selected by setting TSCFG6[4:0] to 0x1~0x4, 0x11, 0x12 or 0x13.

- SMC1= 1'b1 (external clock mode 1). External input ETI is selected as timer clock source.

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source is setting the TSCFG6[4:0] to 0x8. Note that the ETI signal is derived from the ETI pin

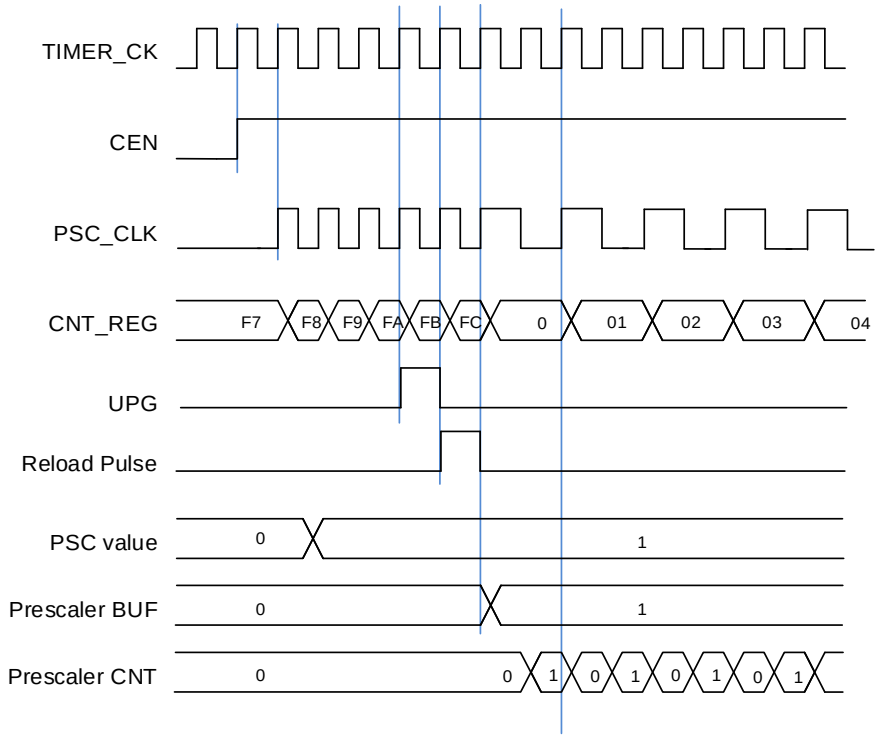
sampled by a digital filter. When the ETI signal is selected as the clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

**Note:** The ETI signal can be input from an external ETI pin or provide by on-chip peripherals, please refer to ***Trigger selection for TIMER0_ETI register (TRIGSEL_TIMER0ETI)*** for more details.

### Clock prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor ranging from 1 to 65536. It is controlled by prescaler register (TIMERx_PSC) which can be changed ongoing, but it is adopted at the next update event.

**Figure 24-3. Counter timing diagram with prescaler division change from 1 to 2**




### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update event will be generated after (TIMERx_CREP0/1+1) times of overflow. Otherwise the update event is generated each time when counter overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up-counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter register, auto reload register, prescaler register) are updated.

*Figure 24-4. Timing diagram of up counting mode, PSC=0/2* and *Figure 24-5. Timing diagram of up counting mode, change TIMERx_CAR ongoing* show some examples of the counter behavior for different clock prescaler factors when TIMERx_CAR=0x99.

**Figure 24-4. Timing diagram of up counting mode, PSC=0/2**

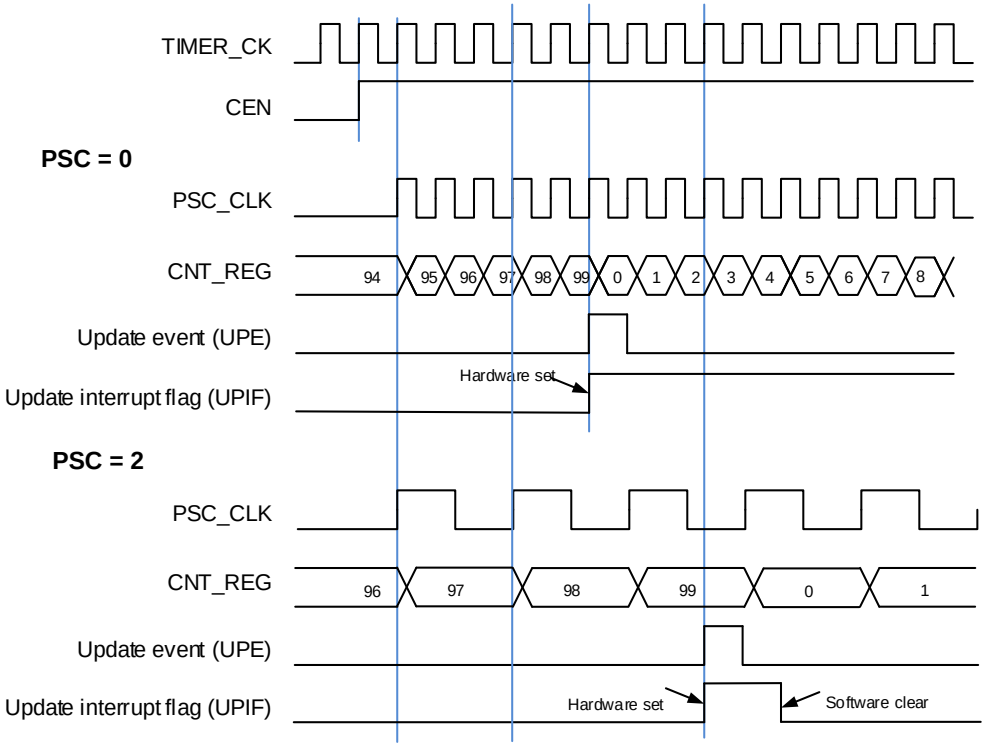
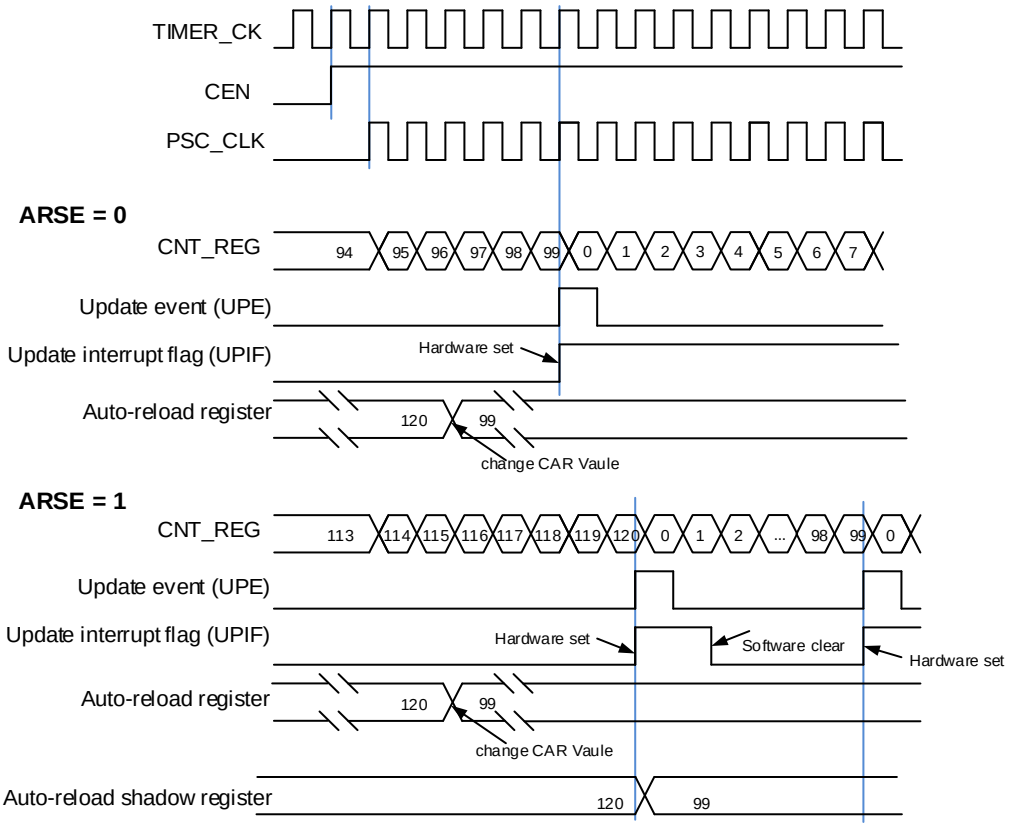


**Figure 24-5. Timing diagram of up counting mode, change TIMERx_CAR ongoing**




## Down counting mode

In this mode, the counter counts down continuously from the counter reload value, which is defined in the TIMERx_CAR register, in a count-down direction. Once the counter reaches 0, the counter restarts to count again from the counter reload value. If the repetition counter is set, the update event will be generated after (TIMERx_CREP0/1+1) times of underflow. Otherwise, the update event is generated each time when counter underflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to the counter reload value and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter register, auto reload register, prescaler register) are updated.

*Figure 24-6. Timing diagram of down counting mode, PSC=0/2* and *Figure 24-7. Timing diagram of down counting mode, change TIMERx_CAR ongoing* show some examples of the counter behavior in different clock frequencies when TIMERx_CAR = 0x99.

**Figure 24-6. Timing diagram of down counting mode, PSC=0/2**

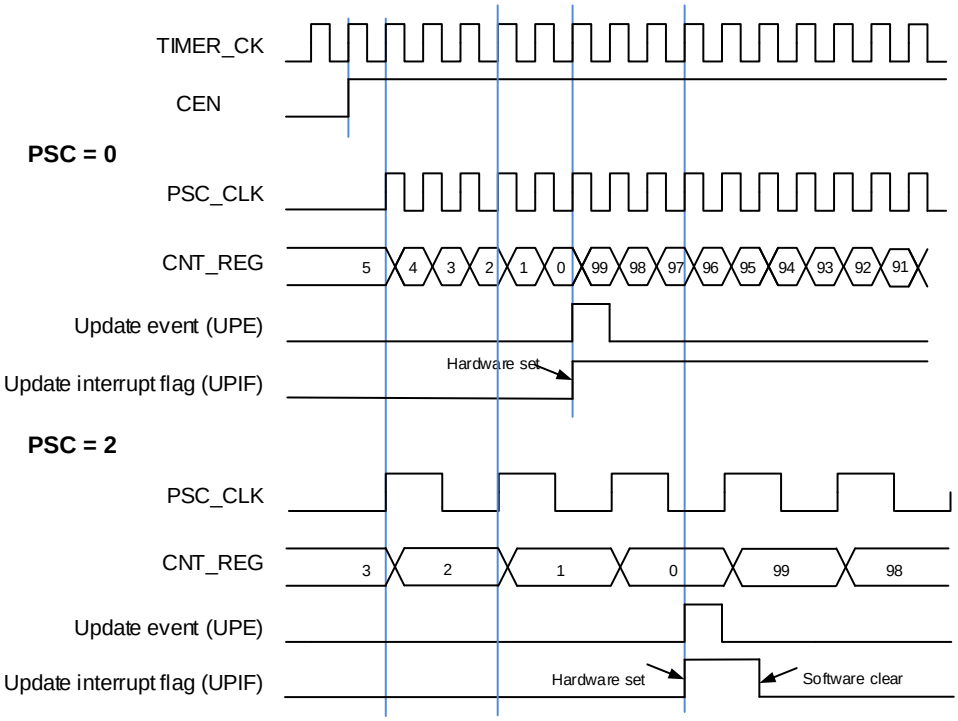
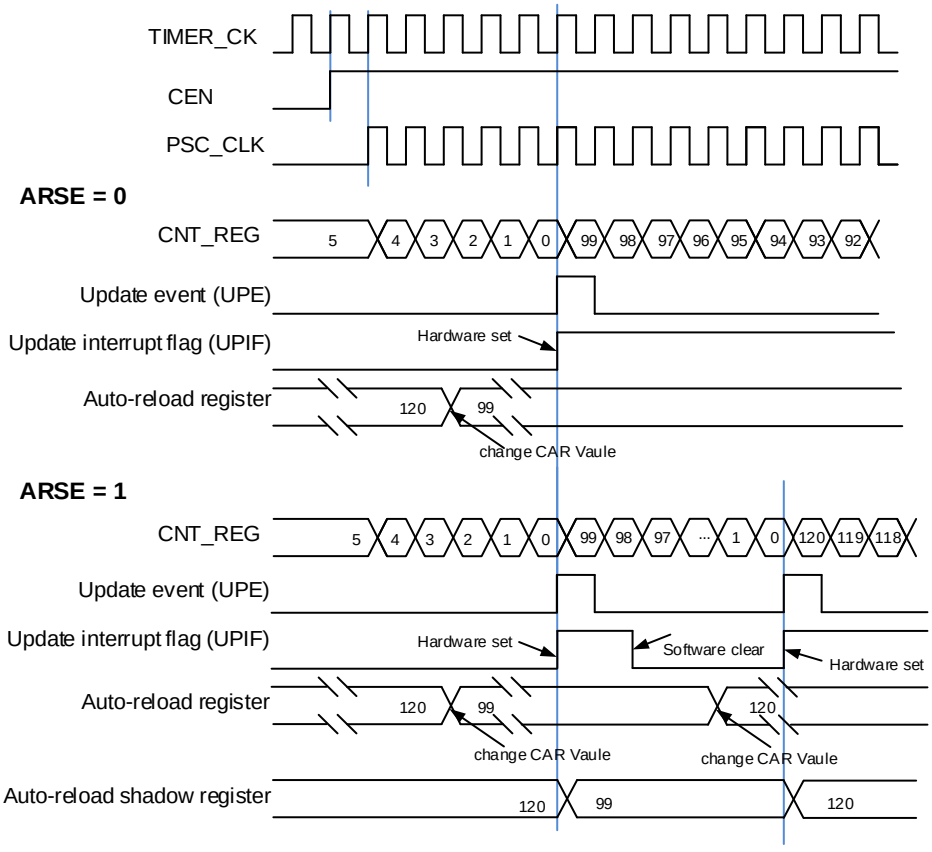



**Figure 24-7. Timing diagram of down counting mode, change TIMERx_CAR ongoing**



### Center-aligned counting mode

In the center-aligned counting mode, the counter counts up from 0 to the counter reload value and then counts down to 0 alternatively. The timer module generates an overflow event when the counter counts to (TIMERx_CAR-1) in the count-up direction and generates an underflow event when the counter counts to 1 in the count-down direction. The counting direction bit DIR in the TIMERx_CTL0 register is read-only and indicates the counting direction when in the center-aligned counting mode. The counting direction is updated by hardware automatically.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 and generate an update event irrespective of whether the counter is counting up or down in the center-aligned counting mode.

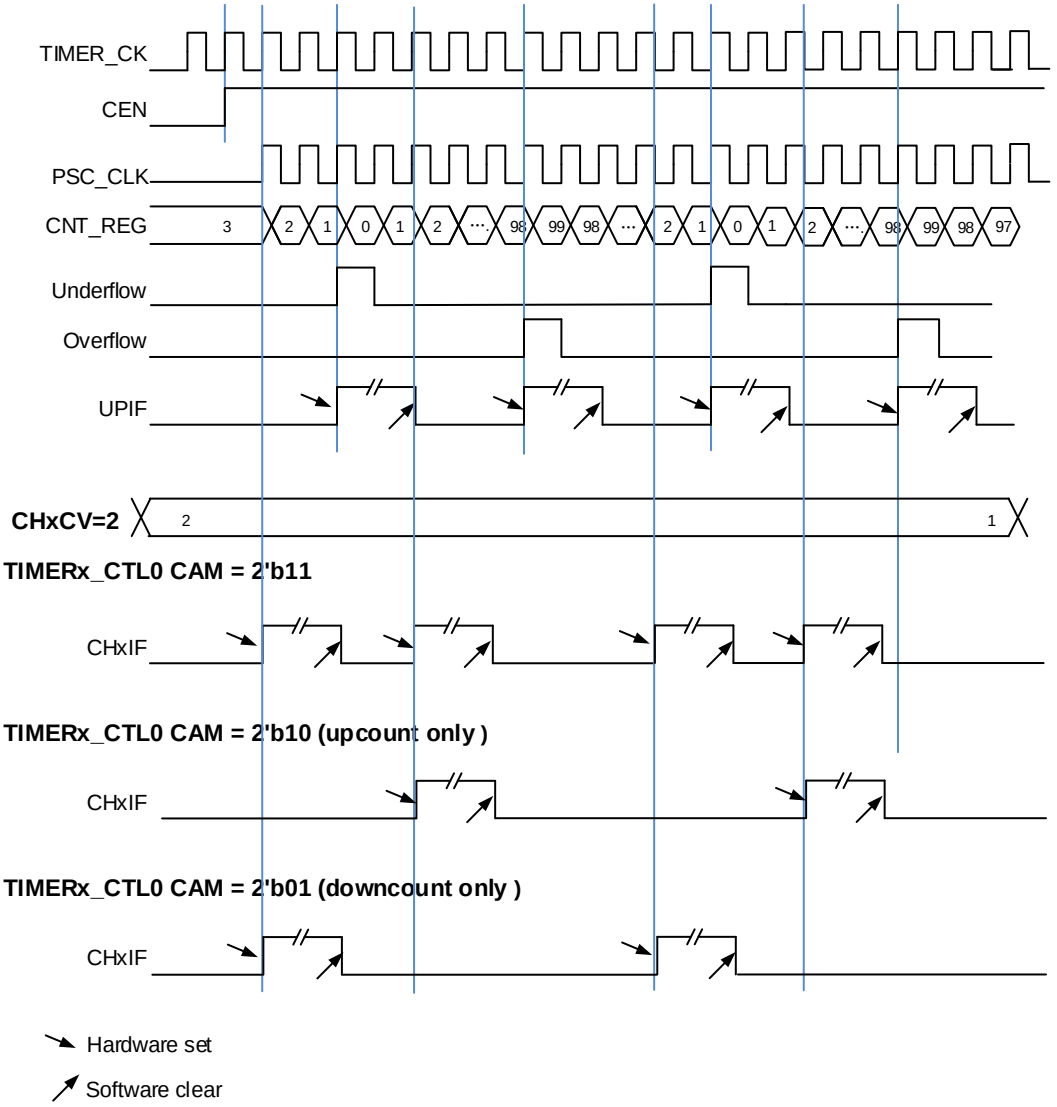
The UPIF bit in the TIMERx_INTF register will be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM[1:0] in TIMERx_CTL0. The details refer to ***Figure 24-8. Timing diagram of center-aligned counting mode***.

If the UPDIS bit in the TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter register, auto-reload register, prescaler register) are updated.

***Figure 24-8. Timing diagram of center-aligned counting mode*** shows some examples of the counter behavior when TIMERx_CAR=0x99. TIMERx_PSC=0x0.

**Figure 24-8. Timing diagram of center-aligned counting mode**




## Counter repetition

The advance timer has two repetitions counter TIMERx_CREP0/1, which can be selected by configuring the CPERSEL bit in the TIMERx_CFG register. The CPEP[7:0] bit-field is 8bits, the CPEP[31:0] bit-field is 32bits and can be read on the fly.

Repetition counter is used to generate the update event or update the timer registers only after a given number (N+1) cycles of the counter, where N is the value of CREP0/1 bit in TIMERx_CREP0/1 register. The repetition counter is decremented at each counter overflow in up counting mode, at each counter underflow in down counting mode or at each counter overflow and at each counter underflow in center-aligned counting mode.

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP0/1 in TIMERx_CREP0/1 register and generate an update event.

The new written CREP0/1 value will not take effect until the next update event. When the

value of CREP0/1 is odd, and the counter is counting in center-aligned mode, the update event is generated (on overflow or underflow) depending on when the written CREP0/1 value takes effect. If an update event is generated by software after writing an odd number to CREP0/1, the update events will be generated on the underflow. If the next update event occurs on overflow after writing an odd number to CREP0/1, then the subsequent update events will be generated on the overflow.

**Figure 24-9. Repetition counter timing diagram of center-aligned counting mode**

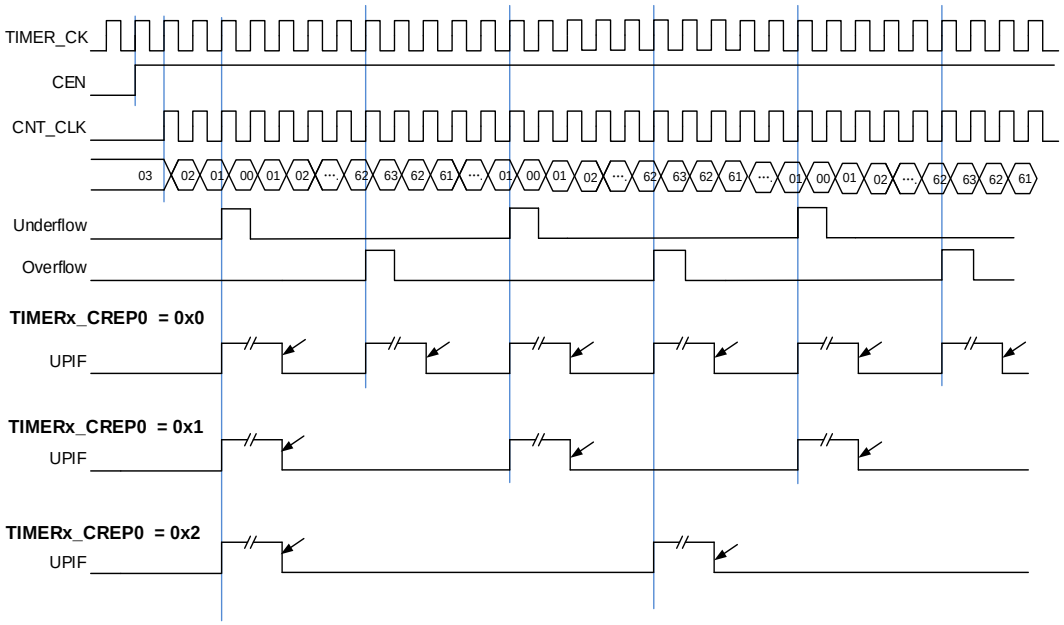



**Figure 24-10. Repetition counter timing diagram of up counting mode**

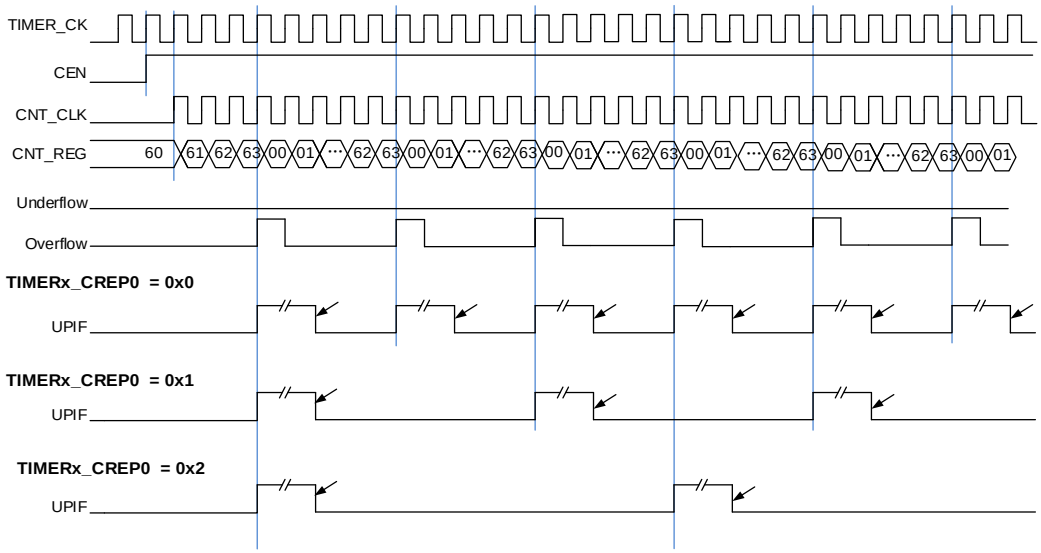
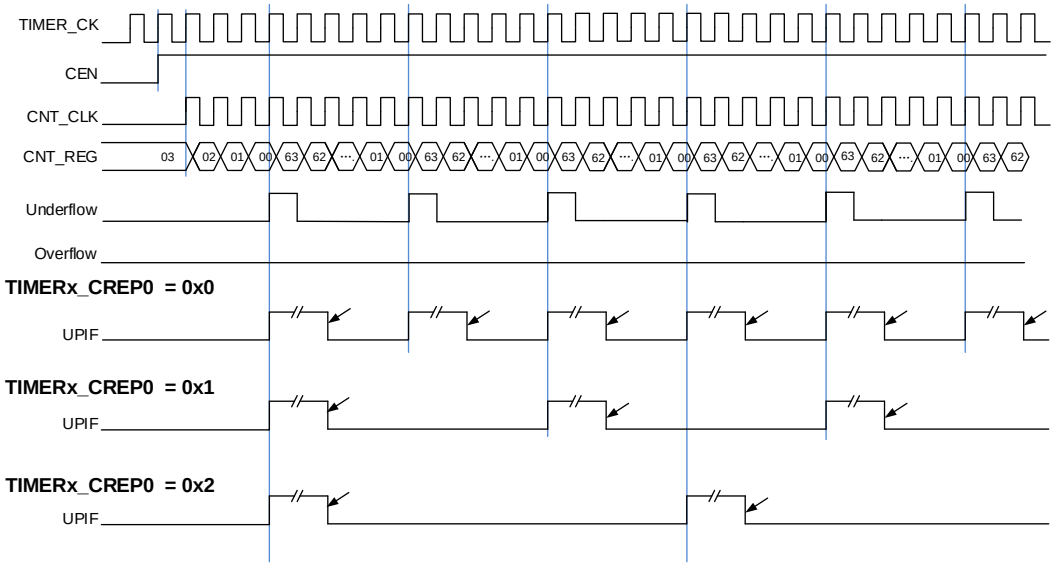


**Figure 24-11. Repetition counter timing diagram of down counting mode**




## Capture/compare channels

The advanced timer has eight independent channels which can be used as capture inputs or compare outputs. Each channel is built around a channel capture compare register including an input stage, a channel controller and an output stage.

When the channels are used for input, channel x and multi mode channel x can perform input capture independently; when the channels are used for comparison output, the channel x and multi mode channel x can output independent and complementary outputs.

- **Input capture mode**

When MCHxMSEL=2'b00(independent mode), channel x and multi mode channel x can perform input capture independently.

Input capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV/ TIMERx_MCHxCV(x=0...3) registers, at the same time the CHxIF/ MCHxIF (x=0..3) bits are set and the channel interrupt is generated if it is enabled when CHxIE/ MCHxIE =1(x=0..3).

**Figure 24-12. Input capture logic for channel 0**

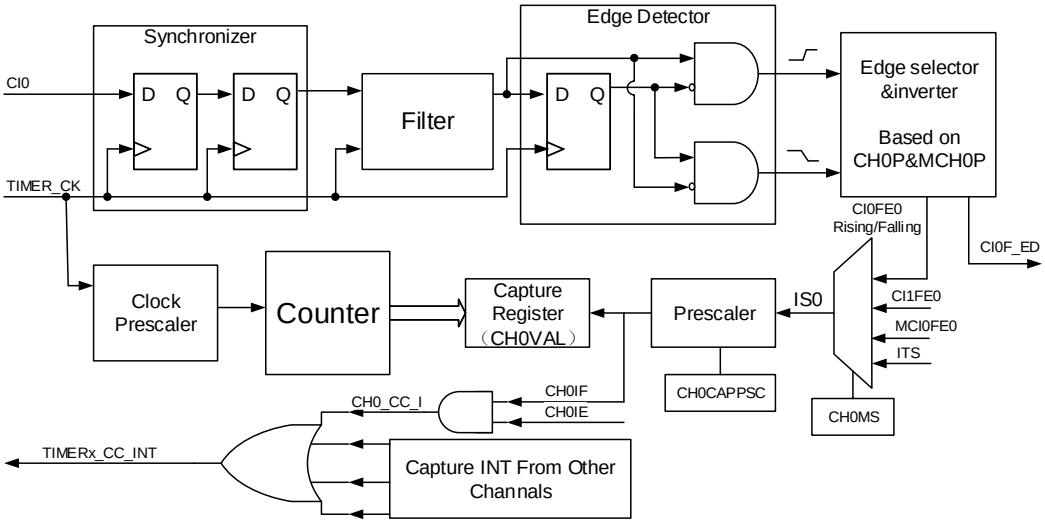
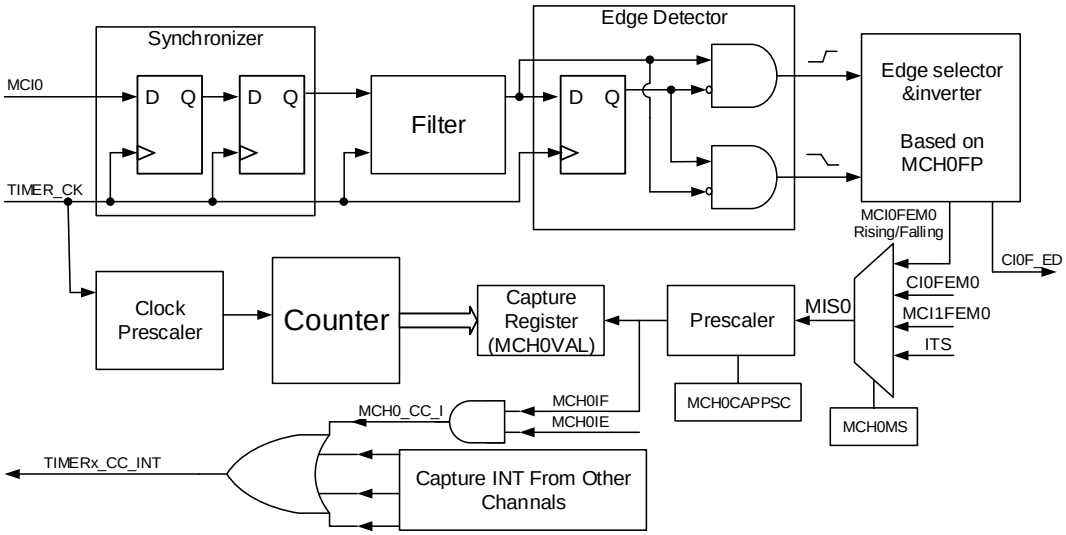



**Figure 24-13. Input capture logic for multi mode channel 0**




The input signals of channelx (CIx/ MCIx) can be the TIMERx_CHx/ TIMERx_MCHxCV signal or the XOR signal of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals (just for CI0).

First, the input signal of channel (CIx/ MCIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP/ MCHxP or MCHxFP bits. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS/ MCHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV/ TIMERx_MCHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT bit in TIMERx_CHCTL0 register and MCHxCAPFLT bit in TIMERx_MCHCTL0 register).

Based on the input signal and quality of requested signal, configure compatible

CHxCAPFLT or MCHxCAPFLT bit.

**Step2**: Edge selection (CHxP and MCHxP bits in TIMERx_CHCTL2 register, MCHxFP[1:0] bits in TIMERx_MCHCTL2 register).

Rising edge or falling edge, choose one by configuring CHxP and MCHxP bits or MCHxFP[1:0] bits.

**Step3**: Capture source selection (CHxMS bit in TIMERx_CHCTL0 register, MCHxMS bit in TIMERx_MCHCTL0 register).

As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x000 or MCHxMS!=0x000) and TIMERx_CHxCV/ TIMERx_MCHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN bits, MCHxIE and MCHxDEN bits in TIMERx_DMAINTEN).

Enable the related interrupt to get the interrupt and DMA request.

**Step5**: Capture enable (CHxEN and MCHxEN bits in TIMERx_CHCTL2).

**Result**: When the wanted input signal is captured, TIMERx_CHxCV/ TIMERx_MCHxCV will be set by counter's value and CHxIF/ MCHxIF bit is asserted. If the CHxIF/ MCHxIF bit is 1, the CHxOF/ MCHxOF bit will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN bits, MCHxIE and MCHxDEN bits in TIMERx_DMAINTEN.

**Direct generation**: A DMA request or interrupt is generated by setting CHxG directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx and TIMERx_MCHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 3'b001 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 3'b010 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty cycle.

## Output compare mode

*Figure 24-14. Output compare logic (when MCHxMSEL = 2'00, x=0, 1, 2, 3)* and *Figure 24-15. Output compare logic (when MCHxMSEL = 2'11, x=0,1,2,3)* show the logic circuit of output compare mode.

**Figure 24-14. Output compare logic (when MCHxMSEL = 2'00, x=0, 1, 2, 3)**

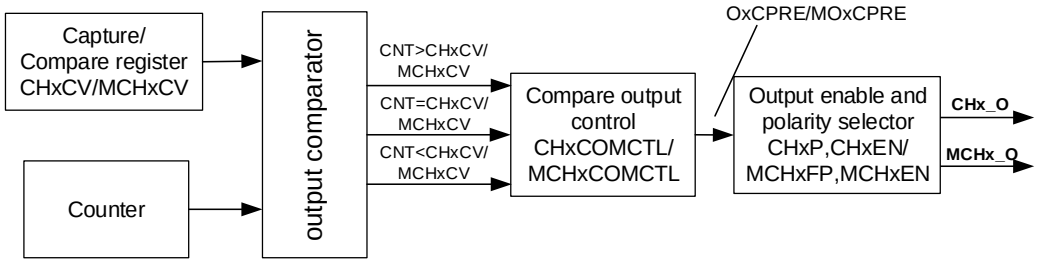
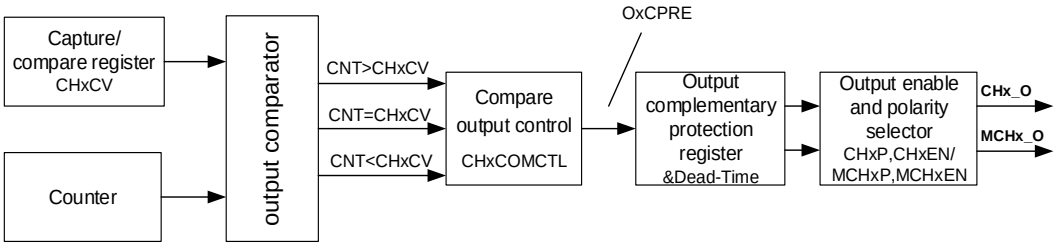


**Figure 24-15. Output compare logic (when MCHxMSEL = 2'11, x=0,1,2,3)**




The relationship between the channel output signal CHx_O/MCHx_O and the OxCPRE/ MOxCPRE signal (more details refer to ***Channel output prepare signal***) is described as blew(the active level of OxCPRE is high and the active level of MOxCPRE is high).

- When MCHxMSEL=2'b00 (in TIMERx_CTL2 register), the MCHx_O output is independent from the CHx_O output. The output level of CHx_O depends on OxCPRE signal, CHxP bit and CHxEN bit (please refer to the TIMERx_CHCTL2 register for more details). The output level of MCHx_O depends on MOxCPRE signal, MCHxFP[1:0] bits and MCHxEN bit (please refer to the TIMERx_MCHCTL2 and TIMERx_CHCTL2 registers for more details). Please refer to ***Figure 24-14. Output compare logic (when MCHxMSEL = 2'00, x=0, 1, 2, 3)***.
- When MCHxMSEL=2'b11, the MCHx_O output is the inverse of the CHx_O output. The output level of CHx_O/MCHx_O depends on OxCPRE signal, CHxP/ MCHxP bits and CHxEN/MCHxEN bits. Please refer to ***Figure 24-15. Output compare logic (when MCHxMSEL = 2'11, x=0,1,2,3)***.

For examples (the MCHx_O output is independent from the CHx_O output):

1) Configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled):
   If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
   If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

2) Configure MCHxP=1 (the active level of MCHx_O is low, contrary to MOxCPRE), MCHxEN=1 (the output of MCHx_O is enabled):
   If the output of MOxCPRE is active(high) level, the output of MCHx_O is active(low) level;
   If the output of MOxCPRE is inactive(low) level, the output of MCHx_O is active(high) level.

When MCHxMSEL=2'b11 and CHx_O and MCHx_O are output at the same time, the specific outputs of CHx_O and MCHx_O are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register. Please refer to ***Outputs Complementary*** for more details.

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV/ TIMERx_MCHxCV register of an output compare channel, the channel (n)

output can be set, cleared, or toggled based on CHxCOMCTL/ MCHxCOMCTL. When the counter reaches the value in the TIMERx_CHxCV/ TIMERx_MCHxCV register, the CHxIF/ MCHxIF bit will be set and the channel (n) interrupt is generated if CHxIE/ MCHxIE = 1. And the DMA request will be asserted, if CHxDEN/ MCHxDEN =1.

So, the process can be divided into several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN/ MCHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL/ MCHxCOMCTL.
- Select the active polarity by CHxP/MCHxP/ MCHxFP.
- Enable the output by CHxEN/ MCHxEN.

**Step3:** Interrupt/DMA request enable configuration by CHxIE/ MCHxIE /CHxDEN/ MCHxDEN.

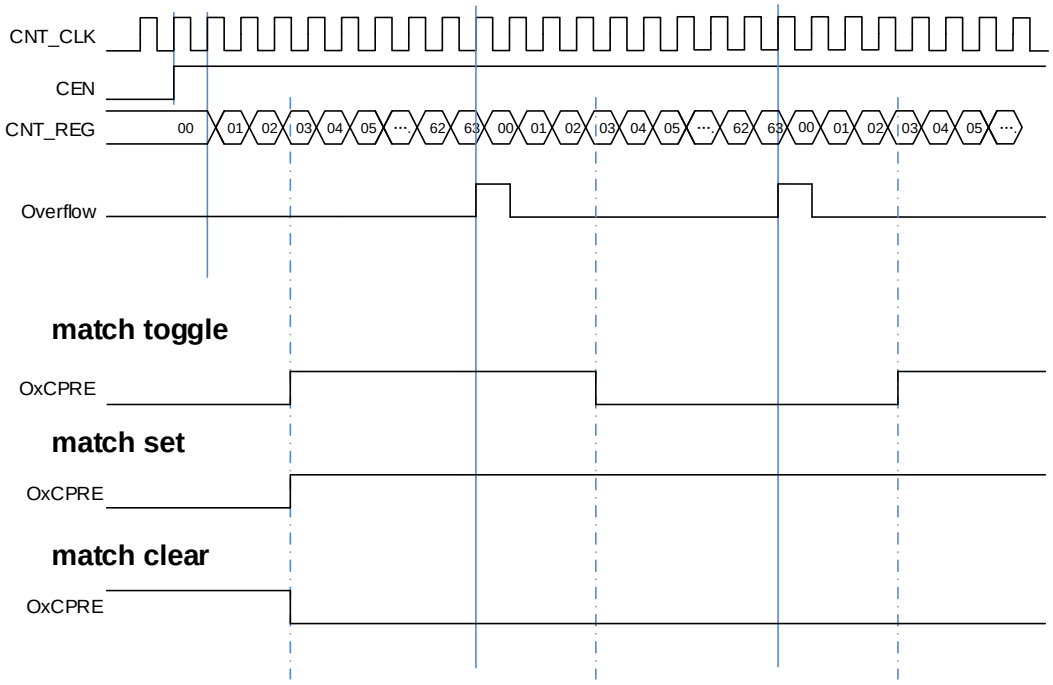
**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV/ TIMERx_MCHxCV.

The TIMERx_CHxCV/ TIMERx_MCHxCV can be changed ongoing to meet the expected waveform.

**Step5:** Start the counter by configuring CEN to 1.

*Figure 24-16. Output-compare in three modes* shows the three compare modes: toggle/set/clear. CARL=0x63, CHxVAL=0x3.

**Figure 24-16. Output-compare in three modes**




## PWM mode

In the PWM output mode (by setting the CHxCOMCTL/ MCHxCOMCTL bit to 4'b0110 (PWM mode 0) or to 4'b0111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV/ TIMERx_MCHxCV registers.

Based on the counter mode, PWM can also be divided into EAPWM (Edge-aligned PWM) and CAPWM (Center-aligned PWM).

The EAPWM's period is determined by TIMERx_CAR and the duty cycle is determined by TIMERx_CHxCV/ TIMERx_MCHxCV. ***Figure 24-17. Timing diagram of EAPWM*** shows the EAPWM output and interrupts waveform.

The CAPWM's period is determined by 2*TIMERx_CAR, and the duty cycle is determined by 2*TIMERx_CHxCV/ TIMERx_MCHxCV. ***Figure 24-18. Timing diagram of CAPWM*** shows the CAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx_CHxCV/ TIMERx_MCHxCV is greater than the value of TIMERx_CAR, the output will be always active in PWM mode 0 (CHxCOMCTL/ MCHxCOMCTL =4'b0110). And if the value of TIMERx_CHxCV/ TIMERx_MCHxCV is greater than the value of TIMERx_CAR, the output will be always inactive in PWM mode 1 (CHxCOMCTL/ MCHxCOMCTL =4'b0111).

**Figure 24-17. Timing diagram of EAPWM**

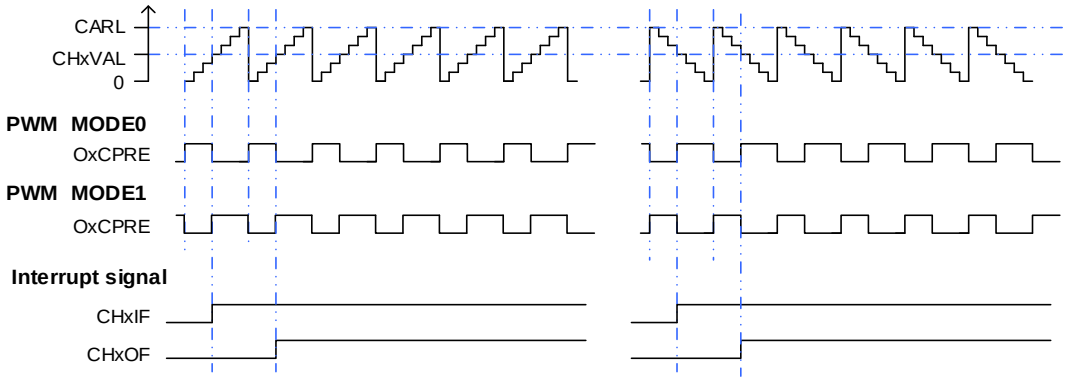
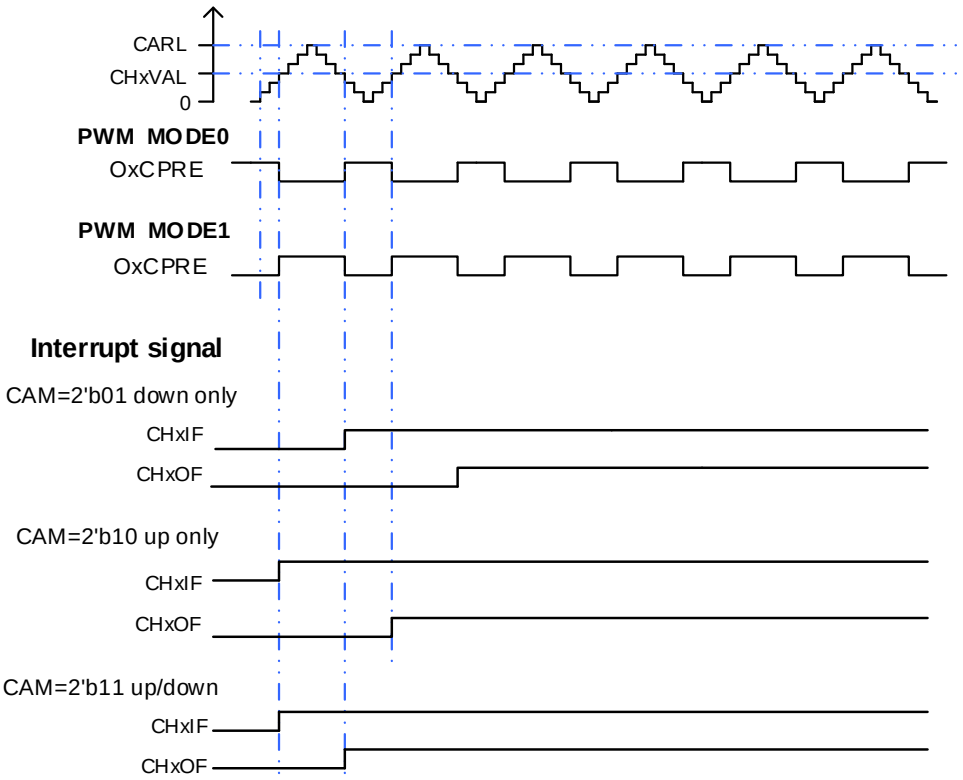


**Figure 24-18. Timing diagram of CAPWM**




### Composite PWM mode

In the Composite PWM mode (CHxCPWMEN = 1'b1, CHxMS[2:0] = 3'b000 and CHxCOMCTL = 4'b0110 or 4'b0111), the PWM signal output in channel x (x=0..3) is composited by CHxVAL and CHxCOMVAL_ADD bits.

If CHxCOMCTL = 4'b0110 (PWM mode 0) and DIR = 1'b0 (up counting mode), or CHxCOMCTL = 4'b0111 (PWM mode 1) and DIR = 1'b1 (Down counting mode), the channel x output is forced low when the counter matches the value of CHxVAL. It is forced high when the counter matches the value of CHxCOMVAL_ADD.

If CHxCOMCTL = 4'b0111 (PWM mode 1) and DIR = 1'b0 (up counting mode), or CHxCOMCTL = 4'b0110 (PWM mode 0) and DIR = 1'b1 (down counting mode) the channel x output is forced high when the counter matches the value of CHxVAL. It is forced low when the counter matches the value of CHxCOMVAL_ADD.

The PWM period is determined by (CARL + 0x0001) and the PWM pulse width is determined by the following table.

**Table 24-3.The Composite PWM pulse width**

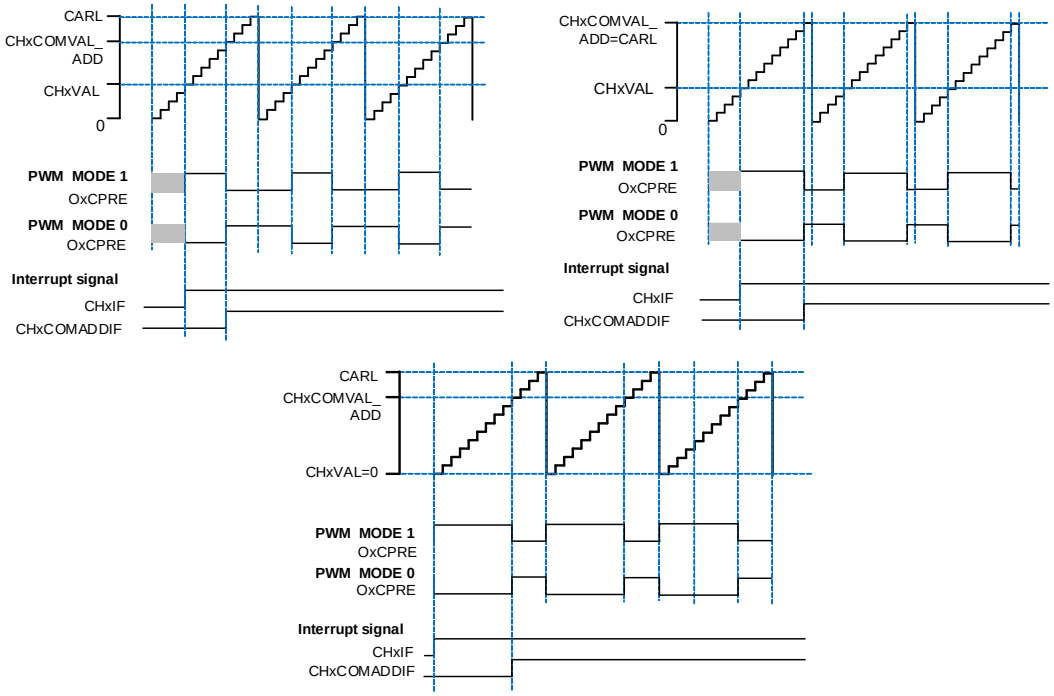
| Condition | Mode | PWM pulse width |
|---|---|---|
| CHxVAL < CHxCOMVAL_ADD ≤ CARL | PWM mode 0 | (CARL + 0x0001) + (CHxVAL – CHxCOMVAL_ADD) |
| | PWM mode 1 | (CHxCOMVAL_ADD – CHxVAL) |
| CHxCOMVAL_ADD < CHxVAL ≤ CARL | PWM mode 0 | (CHxVAL - CHxCOMVAL_ADD) |
| | PWM mode 1 | (CARL + 0x0001) + (CHxCOMVAL_ADD – CHxVAL) |
| (CHxVAL = CHxCOMVAL_ADD ≤ CARL) or (CHxVAL > CARL > CHxCOMVAL_ADD) | PWM mode 0 (up counting) or PWM mode 1 (down counting) | 100% |
| | PWM mode 0 (down counting) or PWM mode 1 (up counting) | 0% |
| CHxCOMVAL_ADD > CARL > CHxVAL | PWM mode 0(up counting) or PWM mode 1(down counting) | 0% |
| | PWM mode 0(down counting) or PWM mode 1(up counting) | 100% |
| (CHxVAL>CARL) and (CHxCOMVAL_ADD > CARL) | - | The output of CHx_O is keeping |

When the counter reaches the value of CHxVAL, the CHxIF bit is set and the channel x interrupt is generated if CHxIE = 1, and the DMA request will be asserted, if CHxDEN=1. When the counter reaches the value of CHxCOMVAL_ADD, the CHxCOMADDIF bit is set (this flag just used in composite PWM mode, when CHxCPWMEN=1) and the channel x additional compare interrupt is generated if CHxCOMADDIE = 1 (Only interrupt is generated, no DMA request is generated).

According to the relationship among CHxVAL, CHxCOMVAL_ADD and CARL, it can be divided into four situations:

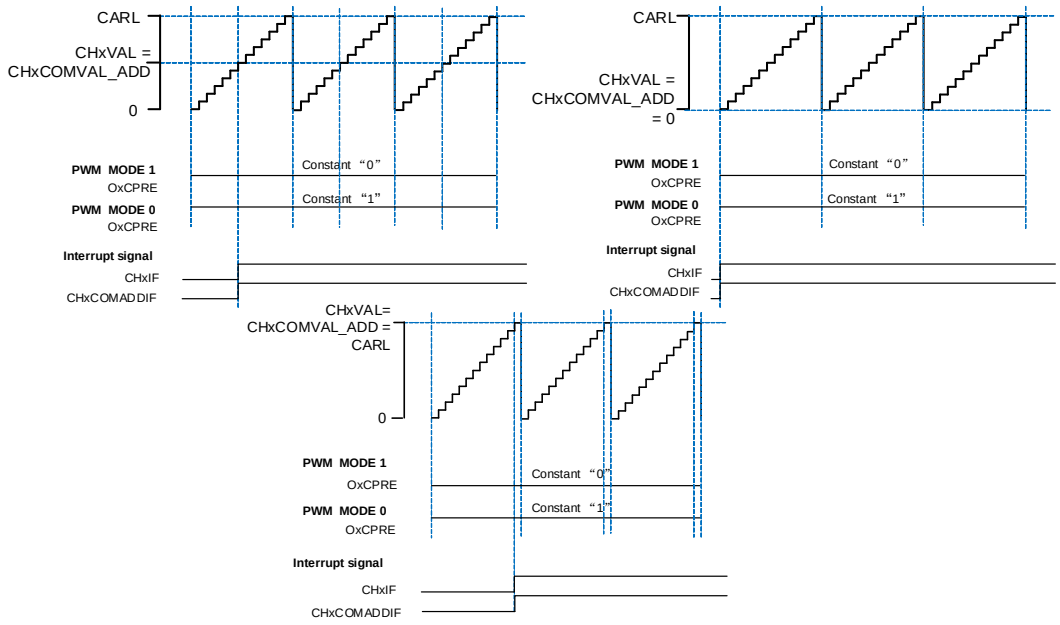
1) CHxVAL < CHxCOMVAL_ADD, and the values of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-19. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD)**



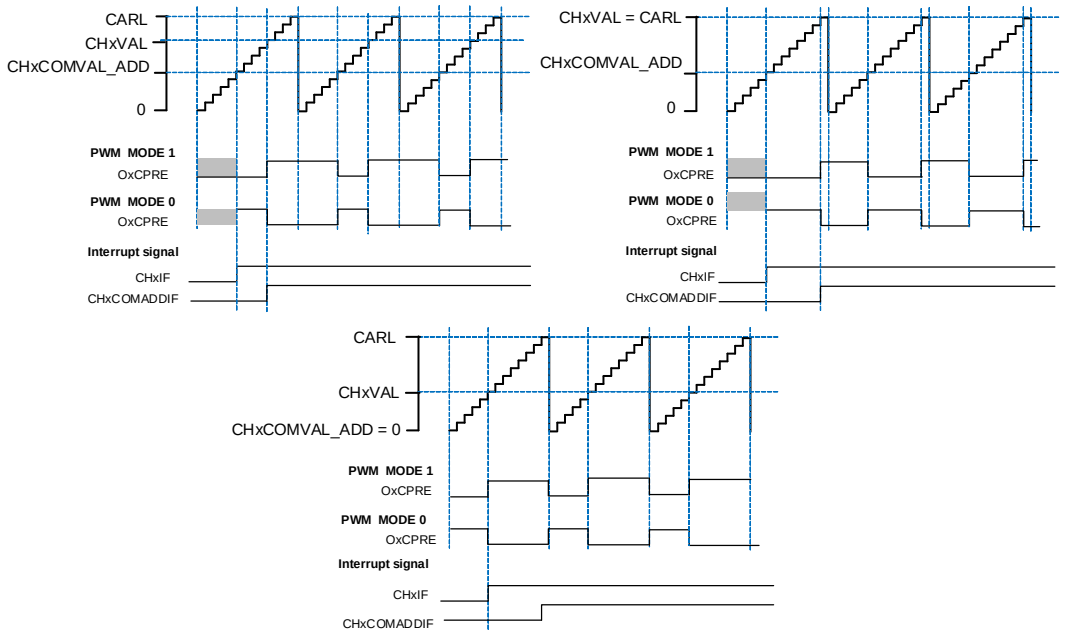

2) CHxVAL = CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-20. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD)**



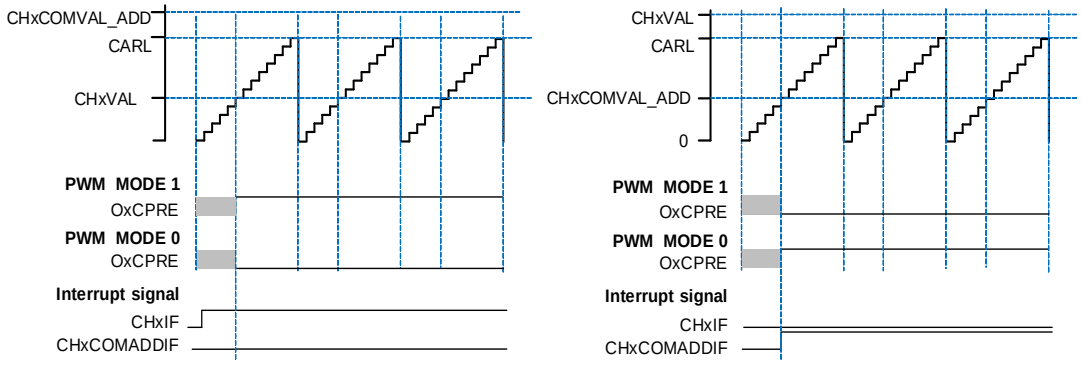

3) CHxVAL > CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-21. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD)**




4) One of the value of CHxVAL and CHxCOMVAL_ADD exceeds CARL.

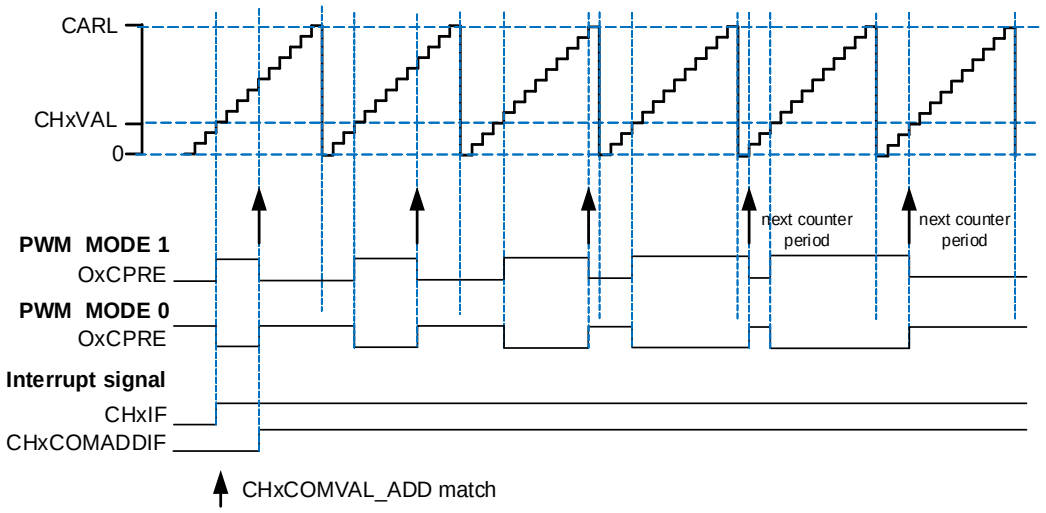
**Figure 24-22. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL**




The composite PWM mode is intended to support the generation of PWM signals where the period is not modified while the signal is being generated, but the duty cycle will be varied. ***Figure 24-23. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD*** shows the PWM output and interrupts waveform.

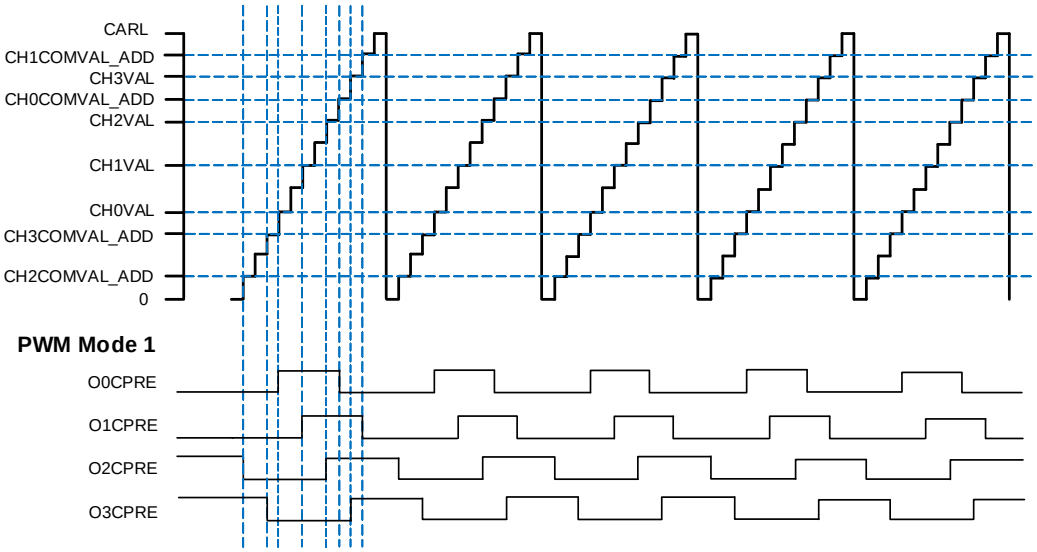
In some cases, the CHxCOMVAL_ADD match can happen on the next counter period (the value of CHxCOMVAL_ADD was written after the counter reaches the value of CHxVAL, and the value of CHxCOMVAL_ADD was less than or equal to the CHxVAL).

**Figure 24-23. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD**




If more than one channels are configured in composite PWM mode, it is possible to fix an offset for the channel x match edge of each pair with respect to other channels. This behavior is useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other pair to help eliminate noise generation. The CHxVAL register value is the shift of the PWM pulse with respect to the beginning of counter period.

**Figure 24-24. Four Channels outputs in Composite PWM mode**




## Output match pulse select

Basing on that CHx_O (x=0..3) outputs are configured by CHxCOMCTL[3:0] (x=0..3) bits when the match events occur, the output signal is configured by CHxOMPSEL[1:0] (x=0..3) bit to be normal or a pulse.

When the match events occur, the CHxOMPSEL[1:0] (x=0..3) bits are used to select the output of OxCPRE which drives CHx_O:

- CHxOMPSEL = 2'b00, the OxCPRE signal is output normally with the configuration of

CHxCOMCTL[3:0] bits;

- CHxOMPSEL = 2'b01, only the counter is counting up, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.
- CHxOMPSEL = 2'b10, only the counter is counting down, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.
- CHxOMPSEL = 2'b11, both the counter is counting up and counting down, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.

**Figure 24-25. CHx_O output with a pulse in edge-aligned mode (CHxOMPSEL≠2'b00)**

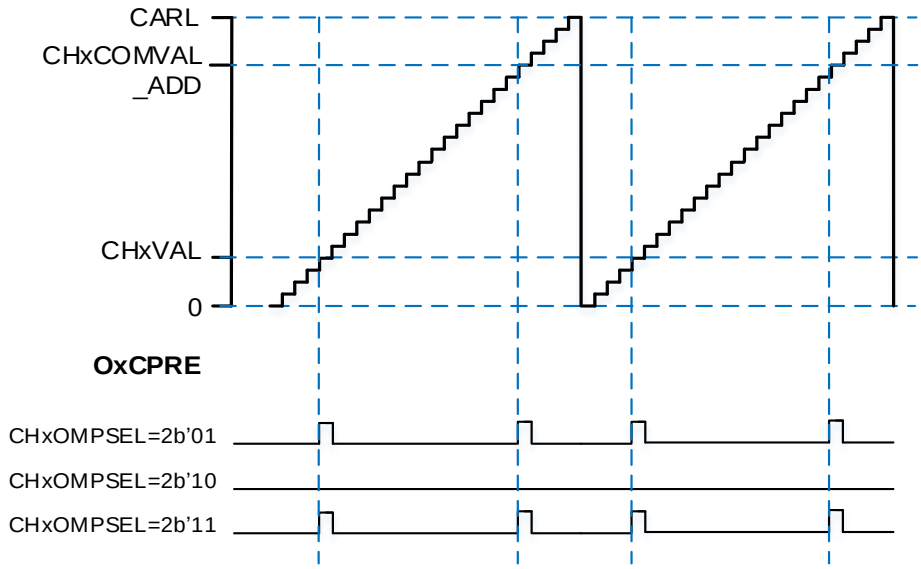
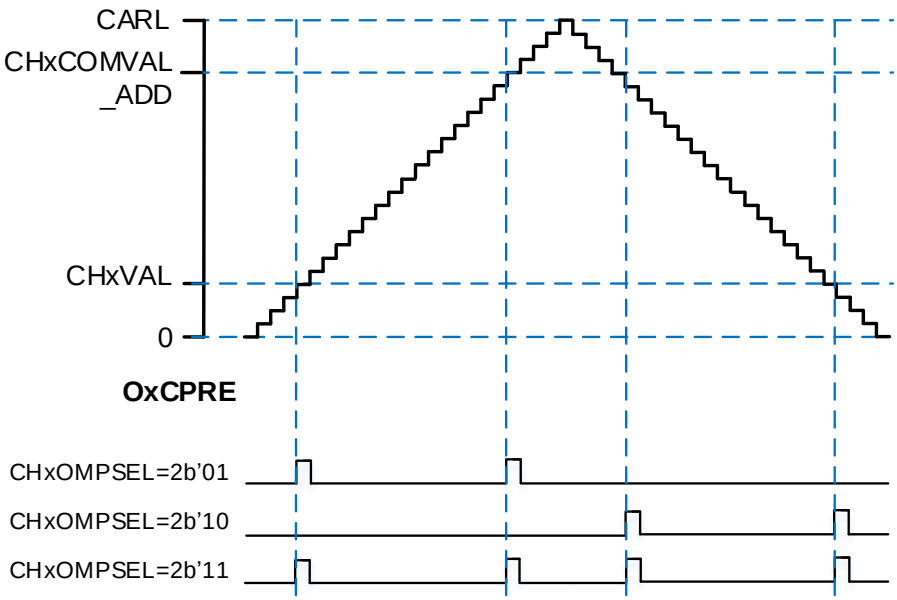



**Figure 24-26. CHx_O output with a pulse in center-aligned mode (CHxOMPSEL≠2'b00)**




### Channel output prepare signal

As is shown in ***Figure 24-14. Output compare logic (when MCHxMSEL = 2'00, x=0, 1, 2, 3)*** and ***Figure 24-15. Output compare logic (when MCHxMSEL = 2'11, x=0,1,2,3)***, when

TIMERx is configured in compare match output mode, a middle signal named OxCPRE or MOxCPRE (channel x output or multi mode channel x output prepare signal) will be generated before the channel outputs signal.

The OxCPRE and MOxCPRE signal have several types of output function. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit and the MOxCPRE signal type is defined by configuring the MCHxCOMCTL bit.

Take OxCPRE as an example for description below, these include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/ PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/ 0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Outputs Complementary

The outputs of CHx_O and MCHx_O have two situations:

- MCHxMSEL=2'b00: The MCHx_O output is independent from the CHx_O output;
- MCHxMSEL=2'b11: The outputs of MCHx_O and CHx_O are complementary and the MCHxCOMCTL bits are not used in the generation of the MCHx_O output.

Function of complementary is for a pair of channels, CHx_O and MCHx_O, the two output signals cannot be active at the same time. The TIMERx has 4 pairs of channels, all the four pairs have this function. The complementary signals CHx_O and MCHx_O are controlled by a group of parameters: the CHxEN and MCHxEN bits in the TIMERx_CHCTL2 register, the POEN, ROS and IOS bits in the TIMERx_CCHP register(when CHx_O and MCHx_O channels has separated deadtime value and break function, please refer to ***Separated dead time insertion and Break function***), ISOx and ISOxN bits in the TIMERx_CTL1 register. The output polarity is determined by CHxP and MCHxP bits in the TIMERx_CHCTL2 register.

When the the outputs of CHx_O and MCHx_O are complementary, there are three situations: output enable、output off-state and output disabled. The details are shown in ***Table 24-4. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)***.

**Table 24-4. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)**

<table>
<tr><th colspan="5">Complementary Parameters</th><th colspan="2">Output Status</th></tr>
<tr><th>POEN</th><th>ROS</th><th>IOS</th><th>CHxEN</th><th>MCHxEN</th><th>CHx_O</th><th>MCHx_O</th></tr>
<tr><td rowspan="4">0</td><td rowspan="4">0/1</td><td rowspan="3">0</td><td rowspan="2">0</td><td>0</td><td colspan="2">CHx_O / CHx_ON = LOW<br>CHx_O / CHx_ON output disable(1).</td></tr>
<tr><td>1</td><td colspan="2" rowspan="3">CHx_O/ CHx_ON output “off-state” (2):<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3)</td></tr>
<tr><td rowspan="2">1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td colspan="2">CHx_O/ CHx_ON output “off-state”:<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN.</td></tr>
<tr><td rowspan="6">1</td><td rowspan="4">0</td><td rowspan="6">0/1</td><td rowspan="2">0</td><td>0</td><td colspan="2">CHx_O/MCHx_O = LOW<br>CHx_O/MCHx_O output disable.</td></tr>
<tr><td>1</td><td>CHx_O = LOW<br>CHx_O output disable.</td><td>MCHx_O=OxCPRE⊕(4)MCHxP<br>MCHx_O output enable.</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable.</td><td>MCHx_O = LOW<br>MCHx_O output disable.</td></tr>
<tr><td>1</td><td>CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable.</td><td>MCHx_O = (! OxCPRE)(5)⊕MCHxP.<br>MCHx_O output enable.</td></tr>
<tr><td rowspan="4">1</td><td rowspan="2">0</td><td>0</td><td>CHx_O = CHxP<br>CHx_O output “off-state”.</td><td>MCHx_O = MCHxP<br>MCHx_O output “off-state”.</td></tr>
<tr><td>1</td><td>CHx_O = CHxP<br>CHx_O output “off-state”</td><td>MCHx_O=OxCPRE⊕MCHxP<br>MCHx_O output enable</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable</td><td>MCHx_O = MCHxP<br>MCHx_O output “off-state”.</td></tr>
<tr><td>1</td><td>CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable</td><td>MCHx_O = (! OxCPRE)⊕MCHxP<br>MCHx_O output enable.</td></tr>
</table>

**Note:**

(1) output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) “off-state”: CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (! OxCPRE)：the complementary output of the OxCPRE signal.

### Dead time insertion

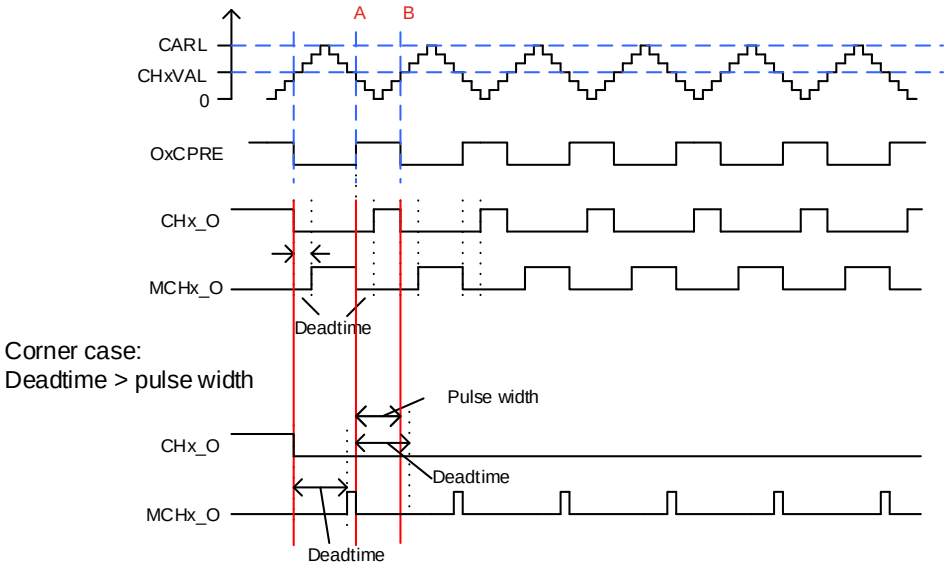
The dead time insertion is enabled when MCHxMSEL=2'b11 and both CHxEN and MCHxEN are configured to 1'b1, it is also necessary to configure POEN to 1. The field named DTCFG defines the dead time delay that can be used for all channels. Refer to the ***Complementary channel protection register (TIMERx_CCHP)*** for details about the delay time.

The dead time delay insertion ensures that two complementary signals are not active at the same time.

When the channel x match event (TIMERx_CNT = CHxVAL) occurs, OxCPRE will be toggled in PWM mode 0. At point A in ***Figure 24-27. Complementary output with dead time insertion***, CHx_O signal remains at the low level until the end of the dead time delay, while MCHx_O signal will be cleared at once. Similarly, at point B when the channelx match event (TIMERx_CNT = CHxVAL) occurs again, OxCPRE is cleared, and CHx_O signal will be cleared at once, while MCHx_O signal remains at the low level until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example: the dead time delay is greater than or equal to the duty cycle of the CHx_O signal, then the CHx_O signal is always inactive (As shown in ***Figure 24-27. Complementary output with dead time insertion***).

**Figure 24-27. Complementary output with dead time insertion**




When CHx_O and MCHx_O channels has separated deadtime value, please refer to ***Separated dead time insertion and Break function***.

By configuring the DTIENCHx (x=0...3) bit in the TIMERx_CTL2 register to realize the independent control of dead-time insertion function for each pair of channels. When the DTIENCHx (x=0...3) bit is "0", the corresponding channels CHx_O and CHx_ON will not be inserted into the dead-time.

### Break function

The MCHx_O output is the inverse of the CHx_O output when the MCHxMSEL=2'b11 (and the MCHxCOMCTL bits are not used in the generation of the MCHx_O output). In this case, CHx_O and MCHx_O signals cannot be set to active level at the same time.

The advanced timers have two kinds of break function: BREAK0 and BREAK1. The break functions can be enabled by setting the BRK0EN and BRK1EN bits in the TIMERx_CCHP register. The break input polarities are configured by the BRK0P and BRK1P bits in TIMERx_CCHP register, the inputs are active on level.

In break functions, CHx_O and MCHx_O are controlled by the POEN, OAEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register.

The break event is the result of logic ORed of all sources. The break functions can handle three types of event sources:

- External sources: coming from BRKINx (x=0...2) inputs;
- System sources: HXTAL stuck event which is generated by Clock Monitor CKM in RCU, LVD lock event, Cortex®-M7 LOCKUP_LOCK event or SRAM parity error event;
- On-chip peripheral events: input by comparator output or HPDF watchdog output.

Break events can also be generated by software using BRK0G or BRK1G bits in the TIMERx_SWEVG register.

Refer to ***Figure 24-28. BREAK0 function logic diagram*** and ***Figure 24-29. BREAK1 function logic diagram***, BRKINx(x=0..2) can select GPIO pins from the TRIGSEL module, which can select by ***Trigger selection for TIMER0 BRKIN register (TRIGSEL_TIMER0BRKIN)*** and ***Trigger selection for TIMER7_BRKIN register (TRIGSEL_TIMER7BRKIN)***.

**Figure 24-28. BREAK0 function logic diagram**

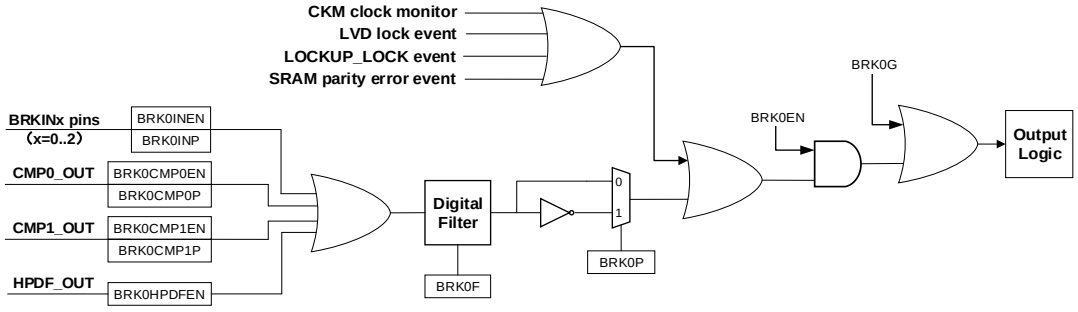
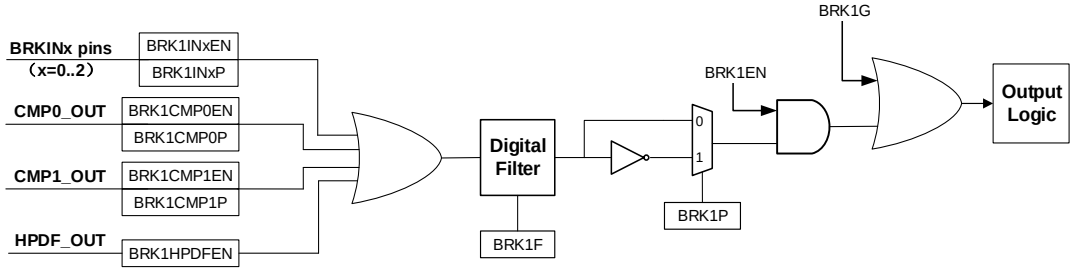


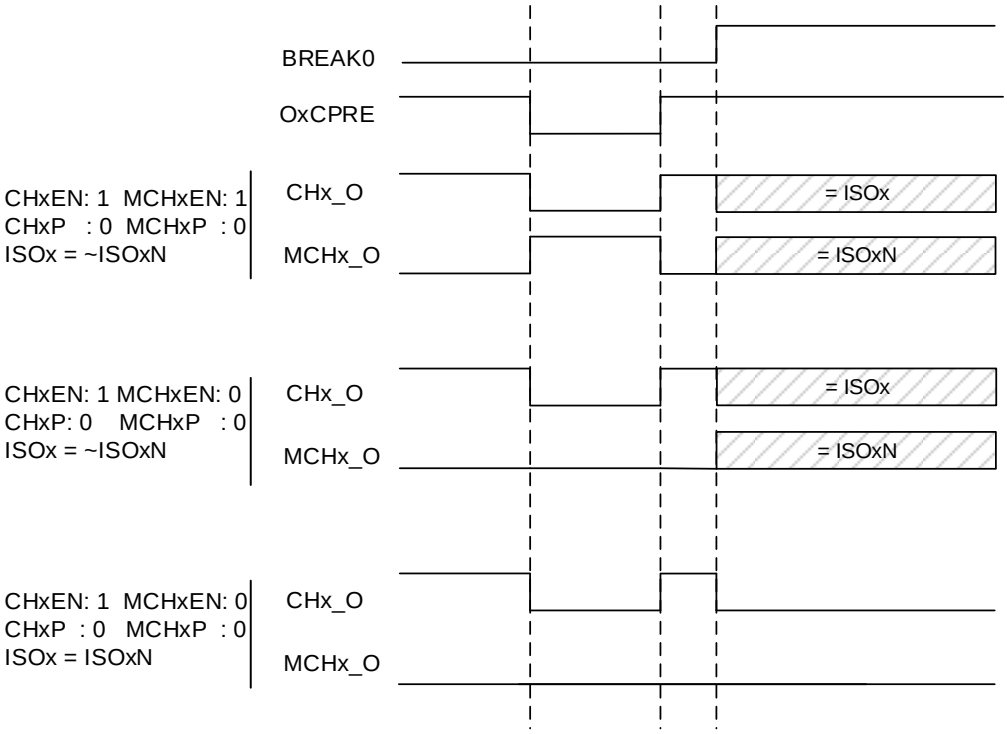

**Figure 24-29. BREAK1 function logic diagram**



BREAK0 can be used to handle the faults of system sources, on-chip peripheral events and external sources. When a BREAK0 event occurs, the outputs are force at an inactive level, or at a predefined level (either active or inactive) after a deadtime duration. BREAK1 only can be used to handle the faults of on-chip peripheral events and external sources. When a BREAK1 event occurs, the outputs are force at an inactive level.

When the MCHxMSEL = 2'b11 and a break occurs, the POEN bit is cleared asynchronously. As soon as POEN is 0, the level of the CHx_O and MCHx_O outputs are determined by the ISOx and ISOxN bits in the TIMERx_CTL1 register. If IOS = 0, the timer releases the enable output, otherwise, the enable output remains high. When IOS=1, the output behavior of the channel is shown in ***Figure 24-30. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1)***. The complementary outputs are first in the reset state, and then the dead time generator is reactivated to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead time.

**Figure 24-30. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1)**




BREAK0 function has a higher priority than BREAK1 function. BREAK1 function only can be used when the IOS =1 and ROS =1.

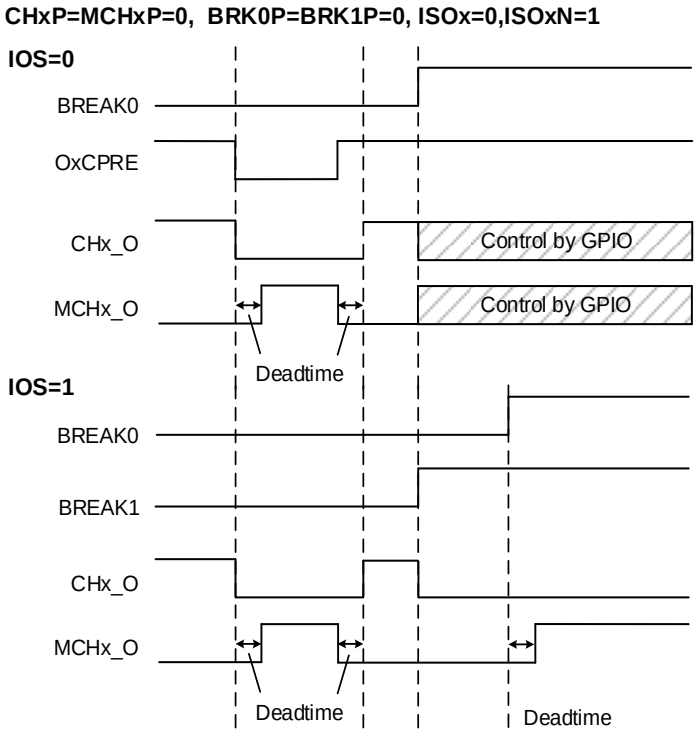
**Table 24-5. Output behavior of the channel in response to a BREAK0 and BREAK1 (the**

**break input is high active)**

| BREAK0 inputs | BREAK1 inputs | Output Status | |
|---|---|---|---|
| | | CHx_O | MCHx_O |
| High | High | IOS=1: CHx_O output is inactive and then output to idle level (by IOSx bit) after a deadtime time. IOS=0: CHx_O output disable (inactive). | IOS=1: MCHx_O output is inactive and then output to level (by IOSxN bit) after a deadtime time. IOS=0: MCHx_O output disable (inactive). |
| | Low | | |
| Low | High | CHx_O output disable (inactive). | MCHx_O output disable (inactive). |

When a break occurs, the BRKIF bit in the TIMERx_INTF register will be set. If BRKIE is 1, an interrupt will be generated.

**Figure 24-31. Output behavior of the channel outputs with the BREAK0 and BREAK1**




When CHx_O and MCHx_O channels has separated break function, please refer to ***Separated dead time insertion and Break function***.

By configuring the BEKENCHx (x=0...3) bit in the TIMERx_CTL2 register to realize the independent control of break function for each pair of channels. When the BEKENCHx(x=0…3) bit is “0” and a break event occurs, the corresponding channels CHx_O and MCHx_O will not be changed and the outputs is keeping.

### Locked break function

The BRKINx(x=0...2) input pins of advanced timer have the locked break function, this

function can be enabled by setting the BRK0LK and BRK1LK bits in the TIMERx_CCHP register.

When the locked break function is enabled, the BRKINx(x=0...2) pins need to be configured to open-drain output mode with low level active (BRK0P/ BRK1P=0 and BRK0INxP/ BRK1INxP=0). When any break source requests occur, the corresponding BRKINx(x=0...2) pin can be forced to low level. If the break input polarity is active high (BRK0P/ BRK1P=1 and BRK0INxP/ BRK1INxP=1), the locked break function is invalid.

When the break function is enabled (the BRK0EN =1 or BRK1EN = 1), the BRKINx(x=0...2) pin can be forced to low level with the BRK0G or BRK1G bits setting to 1 by software.

When the break function is disabled (the BRK0EN =0 or BRK1EN = 0), setting the BRK0G or BRK1G bits will have no effect on the BRKINx(x=0..2) pin. The BRK0F or BRK1F bits will set and the channel outputs will be in a safe state.

The BRKINx(x=0..2) pin can be released by setting the BRK0REL/ BRK1REL bit in the TIMERx_CCHP register. When the break input sources are inactive, the BRK0REL/ BRK1REL bit will cleared by hardware and the BRKINx(x=0..2) pin will restore the locked break function.

In the following two cases, the BRKINx(x=0..2) pin cannot be released:

- Break input sources are active: the BRK0REL/ BRK1REL bit is set to 1 and the BRKINx(x=0..2) pin locked break function is released. The break events is still active, because the break input sources are still active.
- POEN=1: when the channel outputs are enabled, the BRKINx(x=0...2) pin cannot be released even if the BRK0REL/ BRK1REL is set.

**Table 24-6. Break function input pins locked/ released conditions**

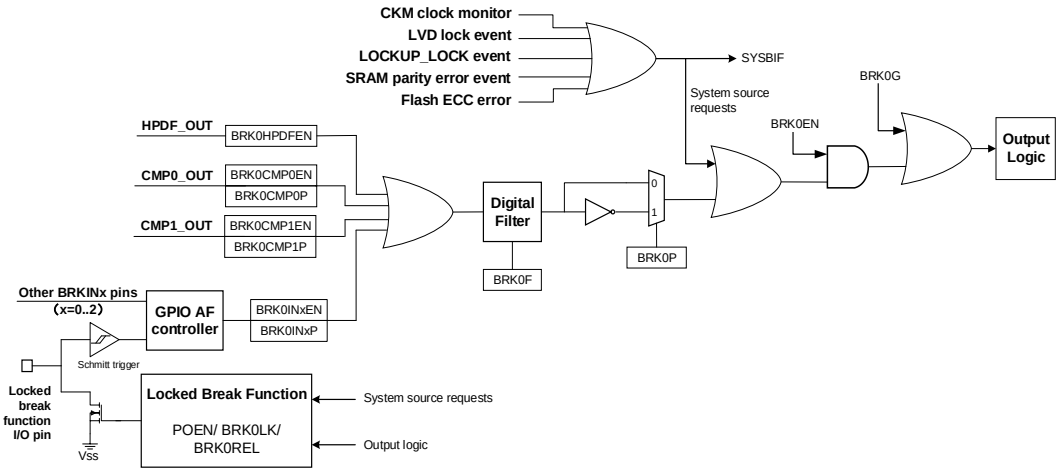
| POEN | BRK0LK/ BRK1LK | BRK0REL/ BRK1REL | Break input pin state |
|---|---|---|---|
| 0 | 1 | 0 | Locked |
| | 1 | 1 | Released |

The locked break function of the BREAK0/BREAK1 input pin BRKINx (x=0...2) is enabled by default (BRK0REL=0 and BRK1REL=0). When the BREAK0/ BREAK1 events occur, the following steps can be used to reconfigure the locked break function:

- Set the BRK0REL or BRK1REL bit to 1 and released the BRKINx (x=0...2) pin;
- The software waits for the system break sources inactive, and then clears the SYSBIF flag；
- The software polls the state of BRK0REL/ BRK1REL bits, until the BRK0REL/ BRK1REL bits are cleared (cleared by hardware).

Then the locked break function of BREAK0/ BREAK1 input pin is re-enabled, and the channel outputs can be restored by setting the POEN bit to 1 by software.

**Figure 24-32. BRKINx (x=0...2) pins logic with BREAK0 function**




## Separated dead time insertion and Break function

The separated dead time insertion and break function for CHx_O and MCHx_O allows that each pair of channels has its own deadtime value and break function. In this function, CHx_O and MCHx_O are actually controlled by the IOS bit、ROS bit and DTCFG[7:0] bits in TIMERx_FCCHPy(y=0..3) register.

By configuring the FCCHPyEN (y=0...3) bits in the TIMERx_FCCHPy(y=0..3) registers can select whether each pair of channels uses the separated dead time insertion and break function. When the FCCHPyEN=0, the ROS、IOS and DTCFG[7:0] bits in TIMERx_CCHP register is active; When the FCCHPyEN=1, the ROS、IOS and DTCFG[7:0] bits in TIMERx_FCCHP0 register is active.

## Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMERx_CH0 and TIMERx_CH1 pins respectively to interact with each other to generate the counter value.

Setting TSCFGy[4:0](y=0..2) != 5'b00000 to select that the counting direction of timer is determined only by the CI0, only by the CI1, or by the CI0 and the CI1.

The DIR bit is modified by hardware automatically during the voltage level change of each direction selection source. The mechanism of changing the counter direction is shown in ***Table 24-7. Counting direction in different quadrature decoder signals***. The CI0FE0 and CI1FE1 are the signals of the CI0 and CI1 after the filtering and polarity selection. The quadrature decoder can be regarded as an external clock with a direction selection. This means that the counter counts continuously from 0 to the counter-reload value. Therefore, users must configure the TIMERx_CAR register before the counter starts to count.

**Table 24-7. Counting direction in different quadrature decoder signals**

| Counting mode | Level | CI0FE0 | | CI1FE1 | |
|---|---|---|---|---|---|
| | | Rising | Falling | Rising | Falling |
| Quadrature decoder mode 0 TSCFG0[4:0]!= 5'b00000 | CI1FE1=1 | Down | Up | - | - |
| | CI1FE1=0 | Up | Down | - | - |
| Quadrature decoder mode 1 TSCFG1[4:0]!= 5'b00000 | CI0FE0=1 | - | - | Up | Down |
| | CI0FE0=0 | - | - | Down | Up |
| Quadrature decoder mode 2 TSCFG2[4:0]!= 5'b00000 | CI1FE1=1 | Down | Up | X | X |
| | CI1FE1=0 | Up | Down | X | X |
| | CI0FE0=1 | X | X | Up | Down |
| | CI0FE0=0 | X | X | Down | Up |

**Note:** "-" means "no counting"; "X" means impossible. "0" means "low level", "1" means "high level".

**Figure 24-33. Example of counter operation in decoder interface mode**

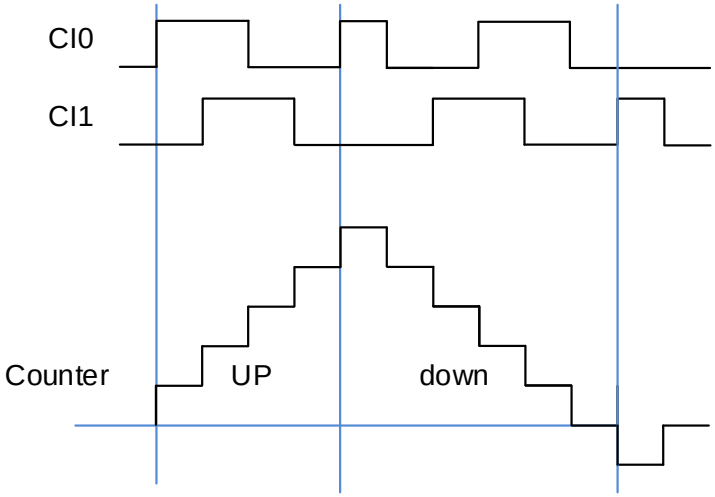
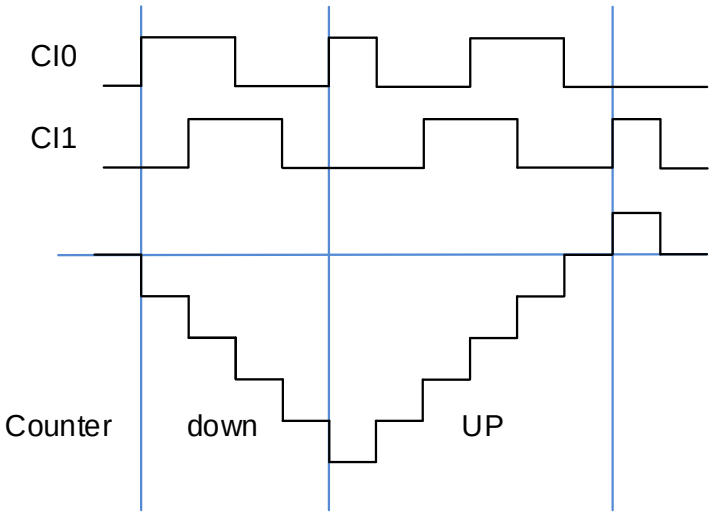



**Figure 24-34. Example of decoder interface mode with CI0FE0 polarity inverted**




### Quadrature decoder signal detection

The quadrature decoder signal jump detection function can be enabled by setting the DECJDEN bit (in TIMERx_CTL2 register) to 1, which can be used to detect whether the level jump edges (rising or falling) of the CI0 and CI1 signals occur at the same time.

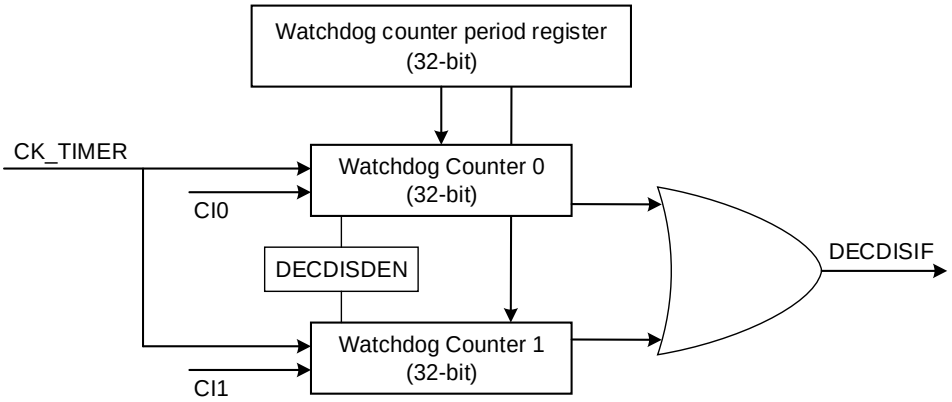
When DECJDEN =1, if the level transitions of the two quadrature signals CI0 and CI1 occur simultaneously, the interrupt flag DECJIF is set, if DECJIE=1, the corresponding interrupt is generated.

The quadrature decoder signal disconnection detection function can be enabled by setting the DECDISDEN bit (in TIMERx_CTL2 register) to 1, which can be used to detect the signal conditions of the CI0 and CI1 signals.

As shown in ***Figure 24-35. Quadrature decoder signal disconnection detection block diagram***. The signal detection module includes two 32-bit watchdog counters and a period register. The CI0 and CI1 signals are used to reset the two watchdog counters respectively.

When DECDISDEN=1, two watchdog counters start counting up at the same time. If the counter continues to count to the watchdog period value (this value is determined by the WDGPER[31:0] bit-field in the TIMERx_WDGPER register), the counter is timeout and the interrupt flag DECDISIF is set. If DECDISIE=1, the corresponding interrupt is generated.

**Figure 24-35. Quadrature decoder signal disconnection detection block diagram**



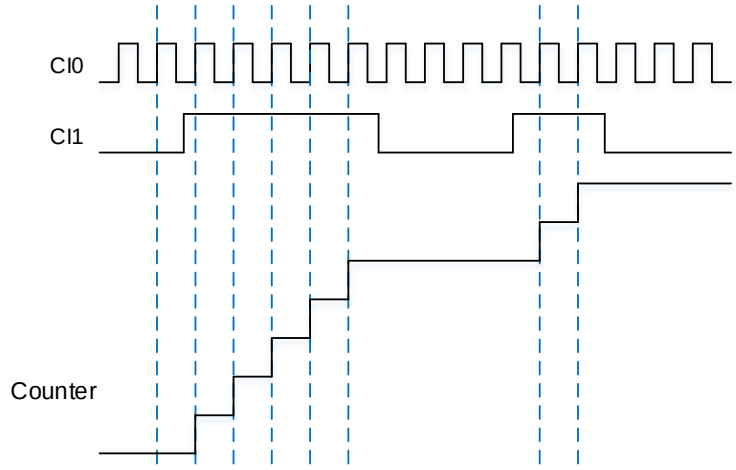

## Non-quadrature decoder

The non-quadrature decoder function has two modes: non-quadrature decoder mode 0 and non-quadrature decoder mode 1, which can be selected by setting TSCFGy[4:0](y=8,9) != 5'b00000.There are two input sources in these two modes: CI0 and CI1.

When the non-quadrature decoder mode 0 is enabled, the CI0 signal is used as the count pulse and CI1 is used as the count selection signal. When CH1P=0, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is high; When CH1P=1, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is low. The more details is shown in ***Figure 24-36. Example of counter operation in non-quadrature decoder mode 0 with CH1P=0***.

**Figure 24-36. Example of counter operation in non-quadrature decoder mode 0 with**

**CH1P=0**

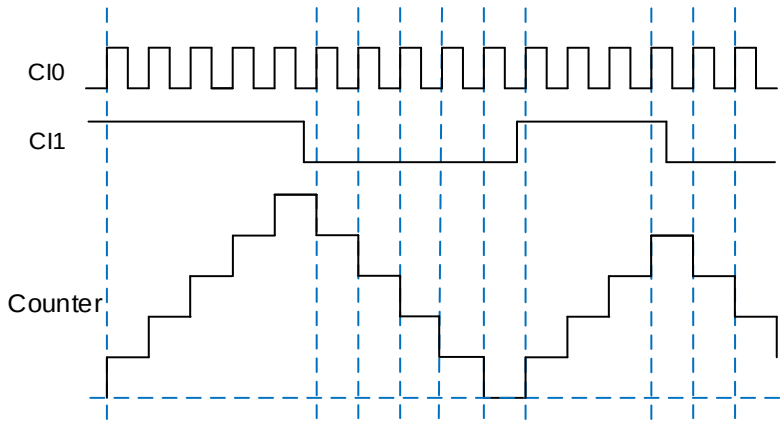



When the non-quadrature decoder mode 1 is enabled, the CI0 signal is used as the count pulse (with the CH0P is used to select the counter edge) and the CI1 signal is used as the count direction selection. The more details is shown in ***Table 24-8. the counter operation in in non-quadrature decoder mode 1*** and ***Figure 24-37. Example of counter operation in non-quadrature decoder mode 1 with CH0P=0***.

**Table 24-8. the counter operation in in non-quadrature decoder mode 1**

| CH0P | level | counter operation |
|---|---|---|
| 0 | CI1 is high | the counter will count up on the rising edge of the CI0 input signal |
| | CI1 is low | the counter will count down on the rising edge of the CI0 input signal |
| 1 | CI1 is high | the counter will count up on the falling edge of the CI0 input signal |
| | CI1 is low | the counter will count down on the falling edge of the CI0 input signal |

**Figure 24-37. Example of counter operation in non-quadrature decoder mode 1 with CH0P=0**



### Hall sensor function

Hall sensor is generally used to control BLDC motor; the advanced timer supports this function.

*Figure 24-38. Hall sensor is used for BLDC motor* shows how to connect the timer and the motor. And two timers are needed. TIMER_in(Advanced/General L0 TIMER）is used to accept three rotor position signals of motor from hall sensors.

Each of the 3 hall sensors provides a pulse which is applied to an input capture pin, then both the speed and position of rotor can be calculated by analyzing the hall sensor signals.

By the internal connection function (TRGO0-ITIx), TIMER_in and TIMER_out can be connected. TIMER_out will generate PWM signals to control the speed of BLDC motor based on the ITIx. Then, the feedback circuit is finished, you can change the configuration to fit your request.

Because the advanced/general L0 TIMER has the input XOR function, they can be used as the TIMER_in timer. And the advanced timer has the functions of complementary output and dead time, so it can be used as the TIMER_out timer. Else, based on the timer's internal connection relationship, pair's timers can be selected. For example:

TIMER_in (TIMER0) -> TIMER_out (TIMER7 ITI0)
TIMER_in (TIMER1) -> TIMER_out (TIMER0 ITI1)

After appropriate interconnected timers are selected and wires are connected, the timers need to be configured. Some key settings are as follows:

- Enable XOR by setting TI0S, then, the change of each input signal will make the CI0 toggle. CH0VAL will record the current value of counter.
- Choose ITIx to trigger commutation by configuring CCUC and CCSE.
- Configure PWM parameters based on the requests.

**Figure 24-38. Hall sensor is used for BLDC motor**

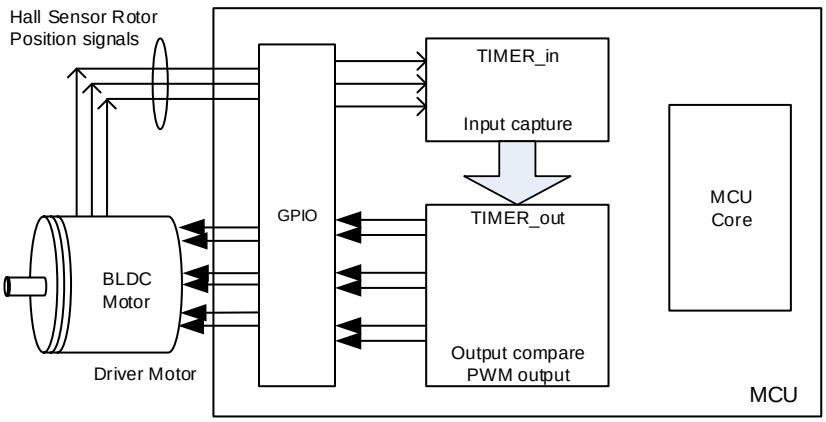
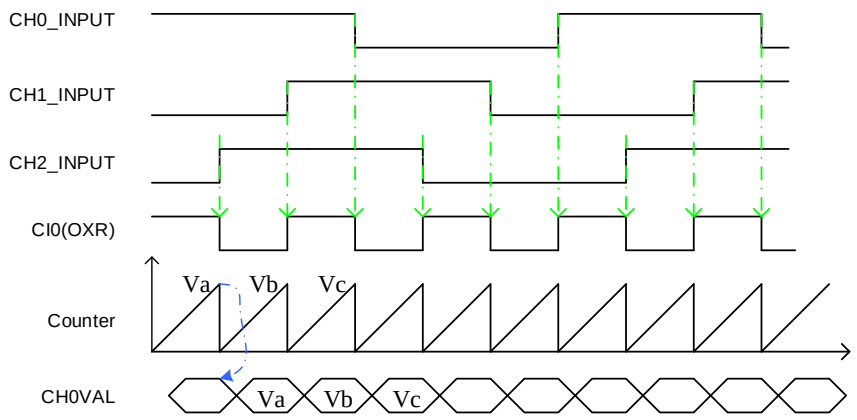
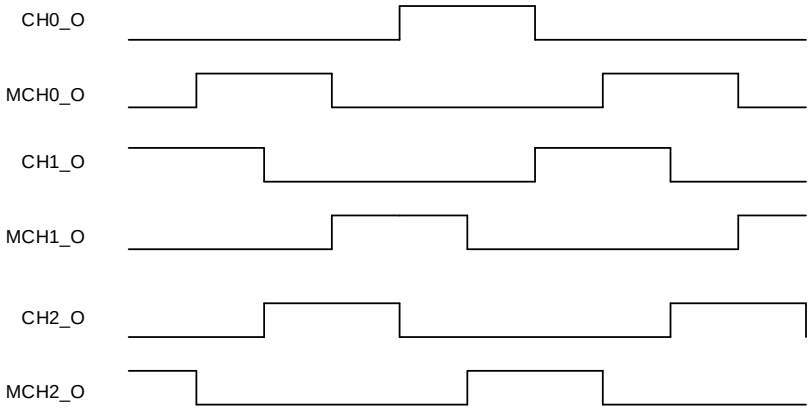


**Figure 24-39. Hall sensor timing between two timers**

**Advanced/General L0 TIMER_in under input capture mode**




**Advanced TIMER_out under output compare mode(PWM with Dead-time)**

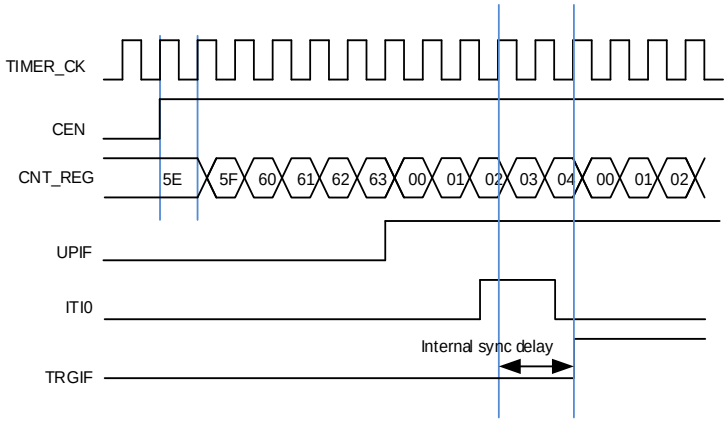
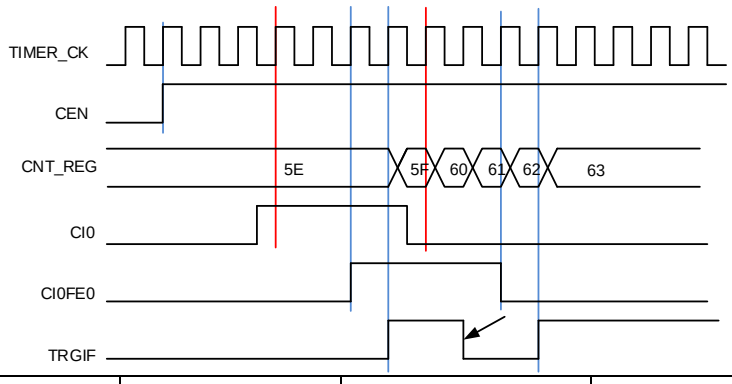



## Master-slave management

The TIMERx can be synchronized with a trigger in several modes including restart mode, pause mode and event mode and so on, which is selected by the TSCFGy[4:0] (y=3..7) in SYSCFG_TIMERxCFG(x=0,7).

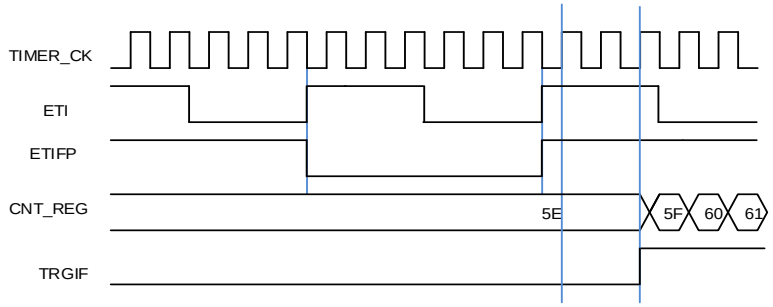
**Table 24-9. Examples of slave mode**

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| **LIST** | TSCFGy[4:0]<br>y=3: restart mode<br>y=4: pause mode<br>y=5: event mode<br>y=6: external clock mode 0<br>y=7: restart + event mode | TSCFGy[4:0]<br>00000:Mode disable<br>00001: ITI0<br>00010: ITI1<br>00011: ITI2<br>00100: ITI3<br>00101: CI0F_ED<br>00110: CI0FE0<br>00111: CI1FE1<br>01000: ETIFP(1) | If CIxFEx(x=0...3) or MCIxFEMx(x=0..3) are selected as the trigger source, configure the CHxP, MCHxP and MCHxFP for the polarity selection and inversion.<br>If ETIFP (the filtered output of external trigger input ETI) is | For the ITIx, no filter and prescaler can be used.<br>For the CIx/ MCIx, filter can be used by configuring CHxCAPFLT/ MCHxCAPFLT, no prescaler can be used.<br>For the ETIFP, filter can be used by |

<table>
<tr><th></th><th>Mode Selection</th><th>Source Selection</th><th>Polarity Selection</th><th>Filter and Prescaler</th></tr>
<tr><td></td><td></td><td>01001: CI2FE2<br>01010: CI3FE3<br>01011: MCI0FEM0<br>01100: MCI1FEM1<br>01101: MCI2FEM2<br>01110: MCI3FEM3<br>10001: ITI12<br>10010: ITI13<br>10011: ITI14</td><td>selected as the trigger source, configure the ETP for polarity selection and inversion.</td><td>configuring ETFC and prescaler can be used by configuring ETPSC.</td></tr>
<tr><td rowspan="2">Exam1</td><td>Restart mode<br>The counter will be cleared and restart when a rising edge of trigger input comes.</td><td>TSCFG3[4:0]<br>5’b00001, ITI0 is selected.</td><td>For ITI0, no polarity selector can be used.</td><td>For the ITI0, no filter and prescaler can be used.</td></tr>
<tr><td colspan="4">Figure 24-40. Restart mode<br> 
</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode<br>The counter will be paused when the trigger input is low, and it will start when the trigger input is high.</td><td>TSCFG4[4:0]<br>=5’b00110,<br>CI0FE0 is selected.</td><td>TI0S=0 (Non-xor) [MCH0P=0, CH0P=0]<br>CI0FE0 does not invert. The capture event will occur on the rising edge only.</td><td>Filter is bypassed in this example.</td></tr>
<tr><td colspan="4">Figure 24-41. Pause mode<br> 
</td></tr>
<tr><td>Exam3</td><td>Event mode</td><td>TSCFG5[4:0]</td><td>ETP = 0, the polarity of</td><td>ETPSC = 1, ETI is</td></tr>
</table>

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| | The counter will start to count when a rising edge of trigger input comes. | =5'b01000, ETIFP is selected. | ETI does not change. | divided by 2. ETFC = 0, ETI does not filter. |

**Figure 24-42. Event mode**




(1) The ETI signal can be input from an external ETI pin or provide by on-chip peripherals, please refer to ***Trigger selection for TIMER0_ETI register (TRIGSEL_TIMER0ETI)*** for more details.

### Single pulse mode

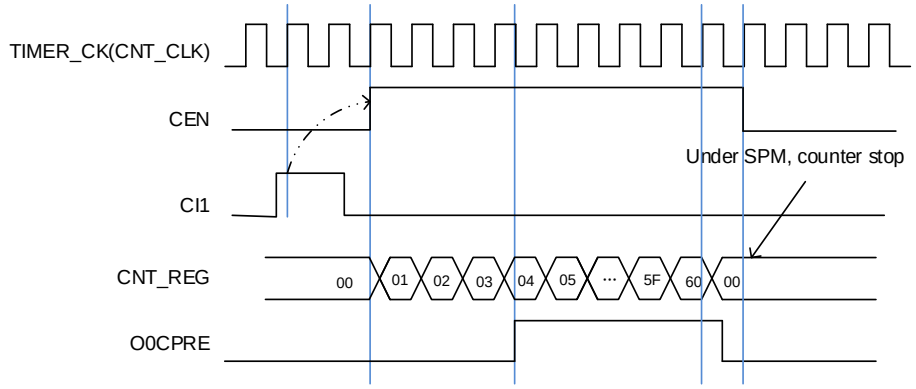
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. If SPM is set, the counter will be cleared and stopped automatically when the next update event occurs. In order to get a pulse waveform, the TIMERx is configured to PWM mode or compare mode by CHxCOMCTL or MCHxCOMCTL bits.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. Setting the CEN bit to 1 or a trigger signal edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 by software, the counter will be stopped and its value will be held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the active edge of trigger which sets the CEN bit to 1 will enable the counter. However, there exists several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE/ MOxCPRE signals will change to, as the compare match event occurs without taking the comparison result into account.

Single pulse mode is also applicable to composite PWM mode (CHxCPWMEN = 1'b1 and CHxMS[2:0] = 3'b000).

**Figure 24-43. Single pulse mode TIMERx_CHxCV=0x04, TIMERx_CAR=0x60**




## Delayable single pulse mode

Delayable single pulse mode is enabled by setting CHxCOMCTL[3:0]/ MCHxCOMCTL[3:0] in TIMERx_CHCTLx/ TIMERx_MCHCTLx registers. In this mode, the pulse width of OxCPRE/ MOxCPRE signal is determined by the TIMERx_CAR register.

Once the timer is set to the delayable single pulse mode, the following configuration is required:

- TIMERx need to work in slave mode and TSCFG7[4:0] != 5'b00000 in SYSCFG_TIMERxCFG(x=0,7) (restart +event mode);
- The CHxCOMCTL[3:0]/ MCHxCOMCTL[3:0] bit-field is setting to 4'b1000 (delayable single pulse mode 0) or 4'b1001 (delayable single pulse mode 1).

In delayable SPM mode 0. The behavior of OxCPRE/ MOxCPRE is performed as in PWM mode 0. When counting up, the OxCPRE/ MOxCPRE is active. When a trigger event occurs, the OxCPRE/ MOxCPRE is inactive. The OxCPRE/ MOxCPRE is active again at the next update event; When counting down, the OxCPRE/ MOxCPRE is inactive, when a trigger event occurs, the OxCPRE/ MOxCPRE is active. The OxCPRE/ MOxCPRE is inactive again at the next update event.

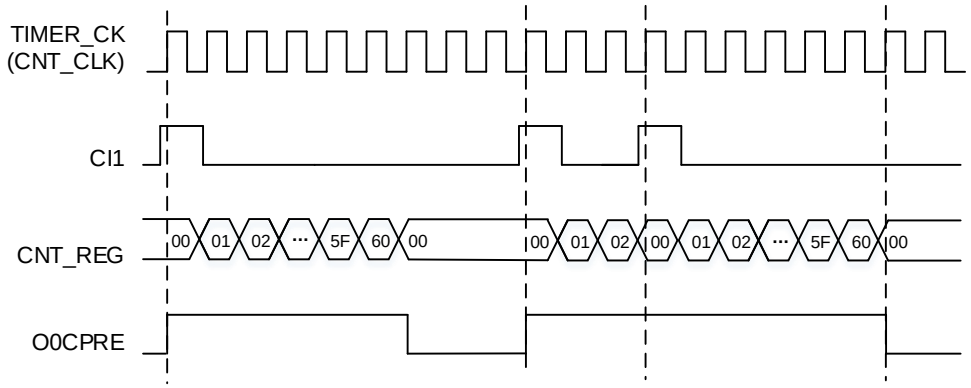
In delayable mode 1. The behavior of OxCPRE/ MOxCPRE is performed as in PWM mode 1. When counting up, the OxCPRE/ MOxCPRE is inactive, when a trigger event occurs, the OxCPRE/ MOxCPRE is active. The OxCPRE/ MOxCPRE is inactive again at the next update event; When counting down, the OxCPRE/ MOxCPRE is active. When a trigger event occurs, the OxCPRE/ MOxCPRE is inactive. The OxCPRE/ MOxCPRE is active again at the next update event.

**Note**:

1) The center-aligned counting mode cannot be used in this mode and the CAM[1:0] = 2'b00( in TIMERx_CTL0 register);
2) When counter counting up (DIR = 0 in TIMERx_CTL0 register), the value of TIMERx_CHxCV/ TIMERx_MCHxCV should be set to 0; When counting down (DIR =1 in TIMERx_CTL0 register), the value of TIMERx_CHxCV/ TIMERx_MCHxCV should be

greater than or equal to the value of TIMERx_CAR register.

**Figure 24-44. delayable single pulse mode with TIMERx_CHxCV=0x00, TIMERx_CAR=0x60**




## Timers interconnection

The timers can be internally connected for timer chaining or synchronization. This can be implemented by configuring one timer to operate in the master mode while configuring another timer to be in the slave mode. The following figures show several examples of trigger selection for the master mode and slave mode.

Some interconnection examples:

- TIMER2 as the prescaler for TIMER0

TIMER2 is configured as a prescaler for TIMER0, steps are shown as follows:

1. Configure TIMER2 in master mode and select its update event (UPE) as trigger output (MMC0=3'b010 in the TIMER2_CTL1 register). Then TIMER2 drives a periodic signal on each counter overflow.
2. Configure TIMER2 period (TIMER2_CAR register).
3. Configure TIMER0 in external clock mode 0 and select the TIMER2 as TIMER0 input trigger source (TRCFG6[4:0] = 5b'00011 in the_SYSCFG_TIMER0CFG1 register).
4. Start TIMER0 by writing '1' to the CEN bit (TIMER0_CTL0 register).
5. Start TIMER2 by writing '1' to the CEN bit (TIMER2_CTL0 register).

- Start TIMER0 with TIMER2's enable/update signal

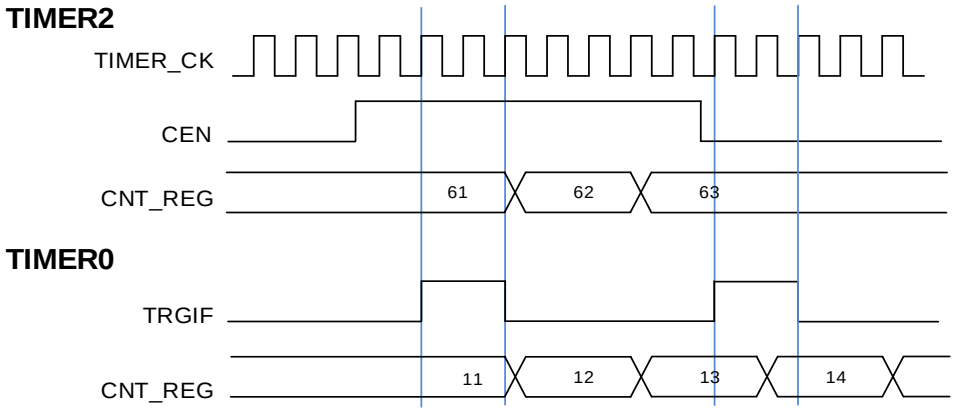
First, enable TIMER0 with the enable signal of TIMER2. Refer to ***Figure 24-45. Trigger mode of TIMER0 controlled by enable signal of TIMER2***. TIMER0 starts counting from its current value with the divided internal clock after being triggered by TIMER2 enable signal output.

When TIMER0 receives the trigger signal, its CEN bit is set automatically and the counter counts until TIMER0 is disabled. Both clock frequency of the counters is divided by 3 from TIMER_CK ($f_{PSC_CLK} = f_{TIMER_CK}$ /3). Steps are shown as follows:

1. Configure TIMER2 in master mode to send its enable signal as trigger output (MMC0=3'b001 in the TIMER2_CTL1 register).

2. Configure TIMER0 in event mode and select the TIMER2 as TIMER0 input trigger source (TRCFG5[4:0] = 5b'00011 in the_SYSCFG_TIMER0CFG0 register).
3. Start TIMER2 by writing 1 to the CEN bit (TIMER2_CTL0 register).

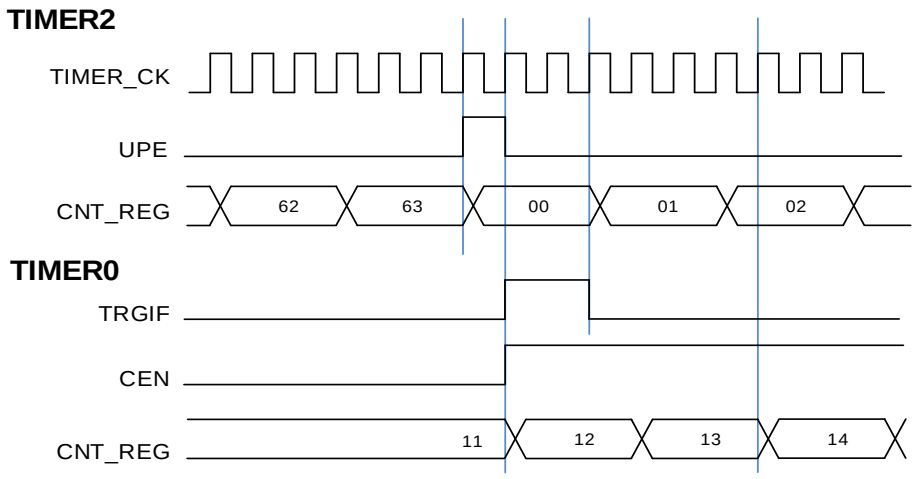
**Figure 24-45. Trigger mode of TIMER0 controlled by enable signal of TIMER2**




In this example, the update event can also be used as trigger source instead of enable signal. Refer to ***Figure 24-46. Trigger mode of TIMER0 controlled by update signal of TIMER2***. Steps are shown as follows:

1. Configure TIMER2 in master mode to send its update event (UPE) as trigger output (MMC0=3'b010 in the TIMER2_CTL1 register).
2. Configure the TIMER2 period (TIMER2_CARL registers).
3. Configure TIMER0 in event mode and select the TIMER2 as TIMER0 input trigger source (TRCFG5[4:0] = 5b'00011 in the_SYSCFG_TIMER0CFG0 register).
4. Start TIMER2 by writing '1' to the CEN bit (TIMER2_CTL0 register).

**Figure 24-46. Trigger mode of TIMER0 controlled by update signal of TIMER2**



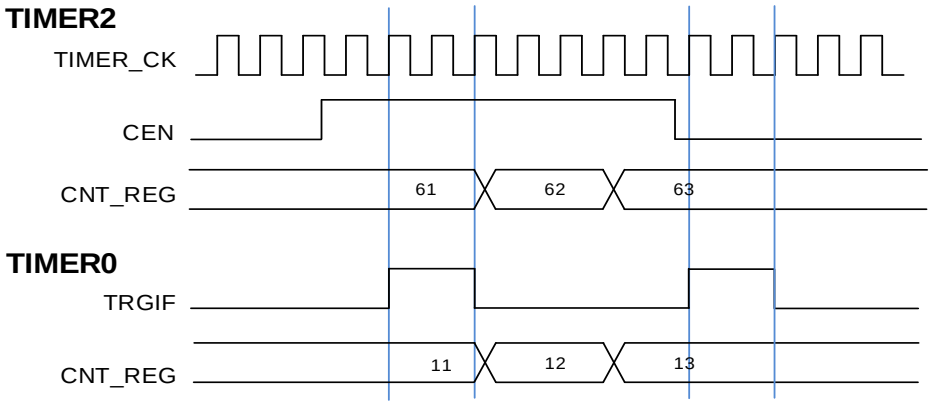

- Enable TIMER0 to count with the enable/O0CPRE signal of TIMER2.

In this example, TIMER0 is enabled with the enable signal of TIMER2. Refer to ***Figure 24-47. Pause mode of TIMER0 controlled by enable signal of TIMER2***. TIMER0 counts with the

divided internal clock only when TIMER2 is enabled. Both clock frequency of the counters is divided by 3 from TIMER_CK ($f_{PSC_CLK} = f_{TIMER_CK}/3$). Steps are shown as follows:

1. Configure TIMER2 in master mode and output enable signal as trigger output (MMC0=3'b001 in the TIMER2_CTL1 register).
2. Configure TIMER0 in pause mode and select the TIMER2 as TIMER0 input trigger source (TRCFG4[4:0] = 5b'00011 in the_SYSCFG_TIMER0CFG0 register).
3. Enable TIMER0 by writing '1' to the CEN bit (TIMER0_CTL0 register).
4. Start TIMER2 by writing '1' to the CEN bit (TIMER2_CTL0 register).
5. Stop TIMER2 by writing '0' to the CEN bit (TIMER2_CTL0 register).

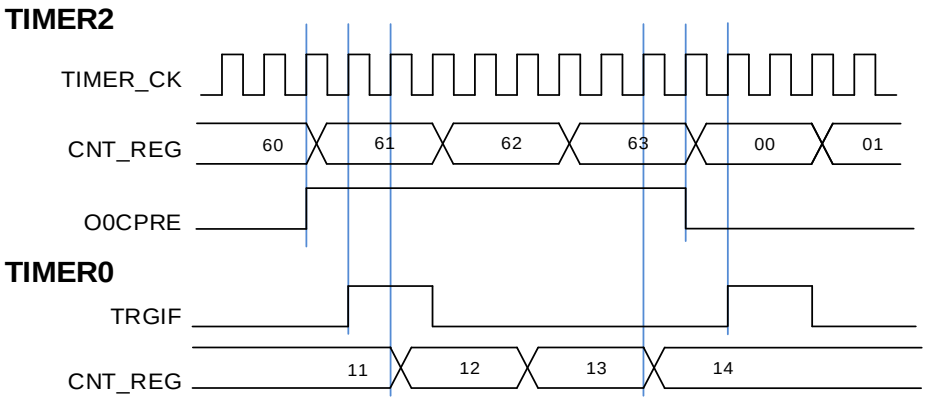
**Figure 24-47. Pause mode of TIMER0 controlled by enable signal of TIMER2**




In this example, O0CPRE can also be used as trigger source instead of enable signal output. Steps are shown as follows:

1. Configure TIMER2 in master mode and O0CPRE as trigger output (MMS=3'b100 in the TIMER2_CTL1 register).
2. Configure the TIMER2 O0CPRE waveform (TIMER2_CHCTL0 register).
3. Configure TIMER0 in pause mode and select the TIMER2 as TIMER0 input trigger source (TRCFG4[4:0] = 5b'00011 in the_SYSCFG_TIMER0CFG0 register).
4. Enable TIMER0 by writing '1' to the CEN bit (TIMER0_CTL0 register).
5. Start TIMER2 by writing '1' to the CEN bit (TIMER2_CTL0 register).

**Figure 24-48. Pause mode of TIMER0 controlled by O0CPRE signal of TIMER2**



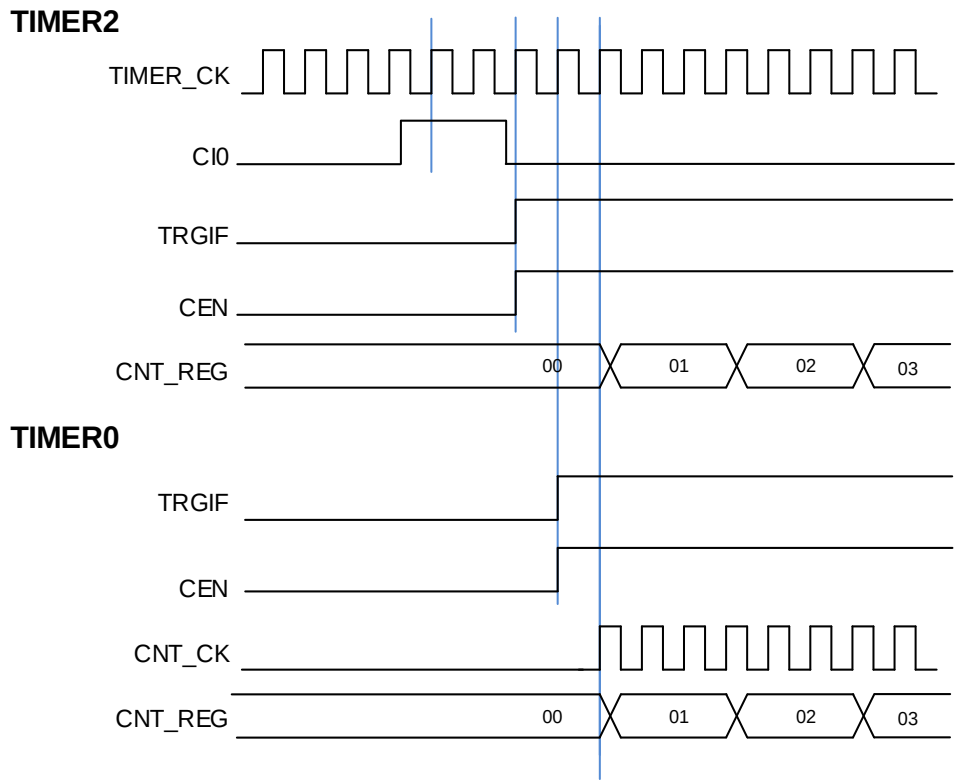
- Using an external trigger to start two timers synchronously.

The start of TIMER0 is triggered by the enable signal of TIMER2, and TIMER2 is triggered by its CI0 input rising edge. To ensure that two timers start synchronously, TIMER2 must be configured in master/slave mode. Steps are shown as follows:

1. Configure TIMER2 in event mode and select the CI0F_ED as TIMER2 input trigger source (TRCFG5[4:0] = 5b'00101 in the_SYSCFG_TIMER2CFG0 register).
2. Configure TIMER2 in master/slave mode by writing MSM=1 (TIMER2_SMCFG register).
3. Configure TIMER0 in event mode and select the TIMER2 as TIMER0 input trigger source (TRCFG5[4:0] = 5b'00011 in the_SYSCFG_TIMER0CFG0 register).

When the CI0 signal of TIMER2 generates a rising edge, two timer counters start counting synchronously with the internal clock and both TRGIF flags are set.

**Figure 24-49. Trigger TIMER0 and TIMER2 by the CI0 signal of TIMER2**




### Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. TIMERx will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of TIMERx_DMATB is configured to PADDR (peripheral base address), then DMA will access the TIMERx_DMATB. In fact, TIMERx_DMATB register is only a buffer, timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0 (1

transfer), the timer sends only one DMA request. While if TIMERx_DMATC is not 0, such as 3 (4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8 and DMATA+0xC at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event asserts, (DMATC+1) times request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

### UPIF bit backup

The UPIF bit backup function is enabled by setting UPIFBUEN in the TIMERx_CTL0 register. The UPIF and UPIFBU bits are fully synchronized and without latency.

By using this function, the UPIF bit in the TIMERx_INTF register will be backuped to the UPIFBU bit in the TIMERx_CNT register. This can avoid conflicts when reading the counter and interrupt processing.

### Timer debug mode

When the Cortex®-M7 is halted, and the TIMERx_HOLD configuration bit in DBG_CTL register is set to 1, the TIMERx counter stops.

### 24.1.5. Registers definition (TIMERx, x=0, 7)

TIMER0 base address: 0x4001 0000

TIMER7 base address: 0x4001 0400

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | UPIFBUEN | Reserved | CKDIV[1:0] | | ARSE | CAM[1:0] | | DIR | SPM | UPS | UPDIS | CEN |
| | | | | rw | | rw | | rw | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | UPIFBUEN | UPIF bit backup enable<br>0: Backup disable. UPIF bit is not backuped to UPIFBU bit in TIMERx_CNT register.<br>1: Backup enabled. UPIF bit is backuped to UPIFBU bit in TIMERx_CNT register. |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division ratio between CK_TIMER (the timer clock) and DTS (the dead time and sampling clock) which is used for the dead time generator and the digital filter.<br>00: $f_{DTS}$= $f_{CK_TIMER}$<br>01: $f_{DTS}$= $f_{CK_TIMER}$ /2<br>10: $f_{DTS}$= $f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:5 | CAM[1:0] | Counter align mode selection<br>00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit.<br>01: Center-aligned and counting down assert mode. The counter counts in center-aligned mode and channel is configured in output mode (CHxMS = 3'b000 in TIMERx_CHCTL0 register). Only when the counter is counting down, compare |

| | | |
|---|---|---|
| | | interrupt flag of channels can be set.<br>10: Center-aligned and counting up assert mode. The counter counts in center-aligned mode and channel is configured in output mode (CHxMS = 3'b000 in TIMERx_CHCTL0 register). Only when the counter is counting up, compare interrupt flag of channels can be set.<br>11: Center-aligned and counting up/down assert mode. The counter counts in center-aligned mode and channel is configured in output mode (CHxMS = 3'b000 in TIMERx_CHCTL0 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set.<br>After the counter is enabled, these bits cannot be switched from 0x00 to non 0x00. |
| 4 | DIR | Direction<br>0: Count up<br>1: Count down<br>This bit is read only when the timer is configured in center-aligned mode or decoder mode. |
| 3 | SPM | Single pulse mode<br>0: Single pulse mode is disabled. Counter continues after an update event.<br>1: Single pulse mode is enabled. The CEN bit is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: Any of the following events generates an update interrupt or a DMA request:<br>– The UPG bit is set.<br>– The counter generates an overflow or underflow event.<br>– The slave mode controller generates an update event.<br>1: Only counter overflow/underflow generates an update interrupt or a DMA request. |
| 1 | UPDIS | Update disable<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. The update event is generated and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set.<br>– The counter generates an overflow or underflow event.<br>– The slave mode controller generates an update event.<br>1: Update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or the slave mode controller generates a hardware reset event. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock mode, pause mode or decoder mode. While in event mode, the hardware can set the CEN |

bit automatically.

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCUC[2:1] | | Reserved | | | | | | | MMC1[2:0] | | | Reserved | | | |
| rw | | | | | | | | | rw | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ISO3N | ISO3 | ISO2N | ISO2 | ISO1N | ISO1 | ISO0N | ISO0 | TI0S | MMC0[2:0] | | | DMAS | CCUC[0] | Reserved | CCSE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | CCUC[2:1] | Commutation control shadow register update control<br>Refer to CCUC [0] description. |
| 29:23 | Reserved | Must be kept at reset value. |
| 22: 20 | MMC1[2:0] | Master mode control 1<br>These bits control the selection of TRGO1 signal.<br>000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO1 pulse occurs. And in the latter case, the signal on TRGO1 is delayed compared to the actual reset.<br>001: Enable. This mode is used to start several timers at the same time or control a slave timer to be enabled in a period. In this mode, the master mode controller selects the counter enable signal as TRGO1. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO1 output, except if the master-slave mode is selected.<br>010: Update. In this mode, the master mode controller selects the update event as TRGO1.<br>011: Capture/compare pulse. In this mode, the master mode controller generates a TRGO1 pulse when a capture or a compare match occurs in channel 0.<br>100: Compare. In this mode, the master mode controller selects the O0CPRE signal as TRGO1.<br>101: Compare. In this mode, the master mode controller selects the O1CPRE signal as TRGO1.<br>110: Compare. In this mode, the master mode controller selects the O2CPRE signal as TRGO1.<br>111: Compare. In this mode, the master mode controller selects the O3CPRE signal as TRGO1.<br>**Note**: The clock of the slave timer or ADC must be enabled prior to receive events |

| | | |
|---|---|---|
| | | from the master timer, and must not be changed on-the-fly while triggers are received from the master timer. |
| 19:16 | Reserved | Must be kept at reset value. |
| 15 | ISO3N | Idle state of multi mode channel 3 complementary output.<br>Refer to ISO0N bit. |
| 14 | ISO3 | Idle state of channel 3 output.<br>Refer to ISO0 bit. |
| 13 | ISO2N | Idle state of multi mode channel 2 complementary output.<br>Refer to ISO0N bit |
| 12 | ISO2 | Idle state of channel 2 output<br>Refer to ISO0 bit |
| 11 | ISO1N | Idle state of multi mode channel 1 complementary output<br>Refer to ISO0N bit |
| 10 | ISO1 | Idle state of channel 1 output<br>Refer to ISO0 bit |
| 9 | ISO0N | Idle state of multi mode channel 0 complementary output<br>0: When POEN bit is reset, MCH0_O is set low.<br>1: When POEN bit is reset, MCH0_O is set high.<br>This bit can be modified only when PROT[1:0] bits in TIMERx_CCHP register is 00. |
| 8 | ISO0 | Idle state of channel 0 output<br>0: When POEN bit is reset, CH0_O is set low.<br>1: When POEN bit is reset, CH0_O is set high.<br>The CH0_O output changes after a dead time if MCH0_O is implemented. This bit can be modified only when PROT[1:0] bits in TIMERx_CCHP register is 00. |
| 7 | TI0S | Channel 0 trigger input selection<br>0: The TIMERx_CH0 pin input is selected as channel 0 trigger input.<br>1: The result of combinational XOR of TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 pins is selected as channel 0 trigger input. |
| 6:4 | MMC0[2:0] | Master mode control 0<br>These bits control the selection of TRGO0 signal, which is sent by master timer to slave timer for synchronization function.<br>000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO0 is delayed compared to the actual reset.<br>001: Enable. This mode is used to start several timers at the same time or control a slave timer to be enabled in a period. In this mode, the master mode controller selects the counter enable signal as TRGO0. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO0 output, except if the |

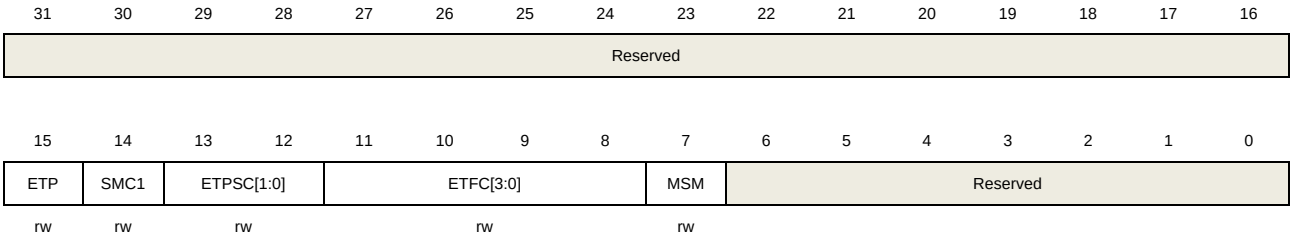
| | | |
|---|---|---|
| | | master-slave mode is selected.<br>010: Update. In this mode, the master mode controller selects the update event as TRGO0.<br>011: Capture/compare pulse. In this mode, the master mode controller generates a TRGO0 pulse when a capture or a compare match occurs in channel 0.<br>100: Compare. In this mode, the master mode controller selects the O0CPRE signal as TRGO0.<br>101: Compare. In this mode, the master mode controller selects the O1CPRE signal as TRGO0.<br>110: Compare. In this mode, the master mode controller selects the O2CPRE signal as TRGO0.<br>111: Compare. In this mode, the master mode controller selects the O3CPRE signal as TRGO0. |
| 3 | DMAS | DMA request source selection<br>0: DMA request of CHx/MCHx is sent when capture/compare event occurs.<br>1: DMA request of channel CHx/MCHx is sent when update event occurs. |
| 2 | CCUC[0] | Commutation control shadow register update control<br>The CCUC[2:1] and CCUC[0] field are used to control the commutation control shadow register update. When the commutation control shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are enabled (CCSE=1), the update control of the shadow registers with the CCUC[2:0] bit-field are shown as below:<br>000: The shadow registers update when CMTG bit is set.<br>001: The shadow registers update when CMTG bit is set or a rising edge of TRGI occurs.<br>100: The shadow registers update when the counter generates an overflow event.<br>101: The shadow registers update when the counter generates an underflow event.<br>110: The shadow registers update when the counter generates an overflow or underflow event.<br>Others: Reserved<br>When a channel does not have a complementary output, this bit has no effect.<br>**Note**: When CCUC[2:0] bit-field are set to 100, 101 and 110, the update of the shadow registers also considers the value the CCUSEL bit in the TIMERx_CFG register. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | CCSE | Commutation control shadow enable<br>0: The shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are disabled.<br>1: The shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are enabled. After these bits have been written, they are updated when commutation event comes.<br>When a channel does not have a complementary output, this bit has no effect. |

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | ETP | External trigger polarity<br>This bit specifies the polarity of ETI signal.<br>0: ETI is active at high level or rising edge.<br>1: ETI is active at low level or falling edge. |
| 14 | SMC1 | Part of slave mode controller is used to enable external clock mode 1<br>In external clock mode 1, the counter is clocked by any active edge of the ETIFP signal.<br>0: External clock mode 1 disabled<br>1: External clock mode 1 enabled<br>It is possible to simultaneously use external clock mode 1 with the restart mode, pause mode or event mode. But the TSCFGy[4:0](y=3,4,5) bits must not be 5b'01000 in this case.<br>The external clock input will be ETIFP if external clock mode 0 and external clock mode 1 are enabled at the same time.<br>**Note:** External clock mode 0 enable is in TSCFG6[4:0] bit-field in SYSCFG_TIMERxCFG1 register. |
| 13:12 | ETPSC[1:0] | External trigger prescaler<br>The frequency of external trigger signal ETIFP must not be higher than 1/4 of TIMER_CK frequency. When the frequency of external trigger signal is high, the prescaler can be enabled to reduce ETIFP frequency.<br>00: Prescaler disabled<br>01: ETIFP frequency divided by 2<br>10: ETIFP frequency divided by 4<br>11: ETIFP frequency divided by 8 |
| 11:8 | ETFC[3:0] | External trigger filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETIFP signal and the length of the digital filter applied to ETIFP. |

| | | |
|---|---|---|
| | | 0000: Filter disabled. $f_{SAMP}$= $f_{DTS}$, N=1.<br>0001: $f_{SAMP}$= $f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}$= $f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}$= $f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}$=$f_{DTS}$/2, N=6.<br>0101: $f_{SAMP}$=$f_{DTS}$/2, N=8.<br>0110: $f_{SAMP}$=$f_{DTS}$/4, N=6.<br>0111: $f_{SAMP}$=$f_{DTS}$/4, N=8.<br>1000: $f_{SAMP}$=$f_{DTS}$/8, N=6.<br>1001: $f_{SAMP}$=$f_{DTS}$/8, N=8.<br>1010: $f_{SAMP}$=$f_{DTS}$/16, N=5.<br>1011: $f_{SAMP}$=$f_{DTS}$/16, N=6.<br>1100: $f_{SAMP}$=$f_{DTS}$/16, N=8.<br>1101: $f_{SAMP}$=$f_{DTS}$/32, N=5.<br>1110: $f_{SAMP}$=$f_{DTS}$/32, N=6.<br>1111: $f_{SAMP}$=$f_{DTS}$/32, N=8. |
| 7 | MSM | Master-slave mode<br>This bit can be used to synchronize the selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected.<br>0: Master-slave mode disabled<br>1: Master-slave mode enabled |
| 6:0 | Reserved | Must be kept at reset value. |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMADDIE | CH2COMADDIE | CH1COMADDIE | CH0COMADDIE | MCH3DEN | MCH2DEN | MCH1DEN | MCH0DEN | MCH3IE | MCH2IE | MCH1IE | MCH0IE | Reserved | | DECDISIE | DECJIE |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | TRGDEN | CMTDEN | CH3DEN | CH2DEN | CH1DEN | CH0DEN | UPDEN | BRKIE | TRGIE | CMTIE | CH3IE | CH2IE | CH1IE | CH0IE | UPIE |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3COMADDIE | Channel 3 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |

| | | |
|---|---|---|
| 30 | CH2COMADDIE | Channel 2 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 29 | CH1COMADDIE | Channel 1 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 28 | CH0COMADDIE | Channel 0 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 27 | MCH3DEN | Multi mode channel 3 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MCH3MSEL[1:0] = 2b'00). |
| 26 | MCH2DEN | Multi mode channel 2 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MCH2MSEL[1:0] = 2b'00). |
| 25 | MCH1DEN | Multi mode channel 1 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH1SEL[1:0] = 2b'00). |
| 24 | MCH0DEN | Multi mode channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH0SEL[1:0] = 2b'00). |
| 23 | MCH3IE | Multi mode channel 3 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MCH3MSEL[1:0] = 2b'00). |
| 22 | MCH2IE | Multi mode channel 2 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |

| | | |
|---|---|---|
| | | **Note:** This bit just used for channel input and output independent mode (when MCH2MSEL[1:0] = 2b'00). |
| 21 | MCH1IE | Multi mode channel 1 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH1SEL[1:0] = 2b'00). |
| 20 | MCH0IE | Multi mode channel 0 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH0SEL[1:0] = 2b'00). |
| 19:18 | Reserved | Must be kept at reset value. |
| 17 | DECDISIE | Quadrature decoder signal disconnection interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for quadrature decoder signal disconnection detection is enabled (when DECDISDEN =1). |
| 16 | DECJIE | Quadrature decoder signal jump (the two signals jump at the same time) interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for quadrature decoder signal jump detection is enabled (when DECJDEN =1). |
| 15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: Disabled<br>1: Enabled |
| 13 | CMTDEN | Commutation DMA request enable<br>0: Disabled<br>1: Enabled |
| 12 | CH3DEN | Channel 3 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 11 | CH2DEN | Channel 2 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 10 | CH1DEN | Channel 1 capture/compare DMA request enable<br>0: Disabled |

| | | |
|---|---|---|
| | | 1: Enabled |
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Disabled<br>1: Enabled |
| 7 | BRKIE | Break interrupt enable<br>0: Disabled<br>1: Enabled |
| 6 | TRGIE | Trigger interrupt enable<br>0: Disabled<br>1: Enabled |
| 5 | CMTIE | Commutation interrupt enable<br>0: Disabled<br>1: Enabled |
| 4 | CH3IE | Channel 3 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 3 | CH2IE | Channel 2 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 2 | CH1IE | Channel 1 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 0 | UPIE | Update interrupt enable<br>0: Disabled<br>1: Enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| CH3COMADDIF | CH2COMADDIF | CH1COMADDIF | CH0COMADDIF | MCH3OF | MCH2OF | MCH1OF | MCH0OF | MCH3IF | MCH2IF | MCH1IF | MCH0IF | Reserved | DECDISIF | DECJIF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | rc_w0 | rc_w0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 2 | 1 | 0 |

| Reserved | SYSBIF | CH3OF | CH2OF | CH1OF | CH0OF | BRK1IF | BRK0IF | TRGIF | CMTIF | CH3IF | CH2IF | CH1IF | CH0IF | UPIF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | . rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3COMADDIF | Channel 3 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 30 | CH2COMADDIF | Channel 2 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 29 | CH1COMADDIF | Channel 1 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 28 | CH0COMADDIF | Channel 0 additional compare interrupt flag.<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No channel 0 output compare interrupt occurred<br>1: Channel 0 output compare interrupt occurred<br>**Note:** This flag just used in composite PWM mode. |
| 27 | MCH3OF | Multi mode channel 3 over capture flag<br>Refer to MCH0OF description. |
| 26 | MCH2OF | Multi mode channel 2 over capture flag<br>Refer to MCH0OF description. |
| 25 | MCH1OF | Multi mode channel 1 over capture flag<br>Refer to MCH0OF description. |
| 24 | MCH0OF | Multi mode channel 0 over capture flag<br>When multi mode channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while MCH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 23 | MCH3IF | Multi mode channel 3 capture/compare interrupt flag<br>Refer to MCH0IF description |
| 22 | MCH2IF | Multi mode channel 2 capture/compare interrupt flag<br>Refer to MCH0IF description |
| 21 | MCH1IF | Multi mode channel 1 capture/compare interrupt flag<br>Refer to MCH0IF description |
| 20 | MCH0IF | Multi mode channel 0 capture/compare interrupt flag |

| | | |
|---|---|---|
| | | This flag is set by hardware and cleared by software.<br>If multi mode channel 0 is in input mode, this flag is set when a capture event occurs.<br>If multi mode channel 0 is in output mode, this flag is set when a compare event occurs.<br>If multi mode channel 0 is set to input mode, this bit will be reset by reading TIMERx_MCH0CV.<br>0: No multi mode channel 0 capture/compare interrupt occurred<br>1: Multi mode channel 0 capture/compare interrupt occurred |
| 19:18 | Reserved | Must be kept at reset value. |
| 17 | DECDISIF | Quadrature decoder signal disconnection interrupt flag<br>0: No quadrature decoder signal disconnection interrupt occurred<br>1: Quadrature decoder signal disconnection interrupt occurred<br>**Note:** This bit just used for quadrature decoder signal disconnection detection is enabled (when DECDISDEN =1). |
| 16 | DECJIF | Quadrature decoder signal jump (the two signals jump at the same time) interrupt flag<br>0: No quadrature decoder signal jump interrupt occurred<br>1: Quadrature decoder signal jump interrupt occurred<br>**Note:** This bit just used for quadrature decoder signal jump detection is enabled (when DECJDEN =1). |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | SYSBIF | System source break interrupt flag<br>This flag is set by hardware when the system sources are active, and cleared by software if the system sources are inactive.<br>0: No system source break interrupt occurred<br>1: System source break interrupt occurred<br>**Note**: When this bit is set, this bit must be cleared by software before the channel outputs are restored. |
| 12 | CH3OF | Channel 3 over capture flag<br>Refer to CH0OF description |
| 11 | CH2OF | Channel 2 over capture flag<br>Refer to CH0OF description |
| 10 | CH1OF | Channel 1 over capture flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred |

| | | |
|---|---|---|
| | | 1: Over capture interrupt occurred |
| 8 | BRK1IF | BREAK1 interrupt flag<br>This flag is set by hardware as soon as the BREAK1 input is active, and cleared by software if the BREAK1 input is not at active level.<br>0: No active level on BREAK1 inputs has been detected.<br>1: An active level on BREAK1 inputs has been detected. An interrupt is generated if BRKIE=1 in the TIMERx_DMAINTEN register. |
| 7 | BRK0IF | BREAK0 interrupt flag<br>This flag is set by hardware when the BREAK0 input is active, and cleared by software if the BREAK0 input is not at active level.<br>0: No active level on break input has been detected.<br>1: An active level on break input has been detected. |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set by hardware on trigger event and cleared by software.<br>When the slave mode controller is enabled in all modes but pause mode, an active edge of trigger input generates a trigger event. When the slave mode controller is enabled in pause mode, either edge of the trigger input can generate a trigger event.<br>0: No trigger event occurred<br>1: Trigger interrupt occurred |
| 5 | CMTIF | Channel commutation interrupt flag<br>This flag is set by hardware when the commutation event of channel occurs, and cleared by software.<br>0: No channel commutation interrupt occurred<br>1: Channel commutation interrupt occurred |
| 4 | CH3IF | Channel 3 capture/compare interrupt flag<br>Refer to CH0IF description |
| 3 | CH2IF | Channel 2 capture/compare interrupt flag<br>Refer to CH0IF description |
| 2 | CH1IF | Channel 1 capture/compare interrupt flag<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 capture/compare interrupt flag<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in input mode, this flag is set when a capture event occurs. If channel 0 is in output mode, this flag is set when a compare event occurs.<br>If channel 0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV.<br>0: No channel 0 interrupt occurred<br>1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware when an update event occurs and cleared by software.<br>0: No update interrupt occurred |

1: Update interrupt occurred

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMADDG | CH2COMADDG | CH1COMADDG | CH0COMADDG | Reserved | | | | MCH3G | MCH2G | MCH1G | MCH0G | Reserved | | | |
| w | w | w | w | | | | | w | w | w | w | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | BRK1G | BRK0G | TRGG | CMTG | CH3G | CH2G | CH1G | CH0G | UPG |
| | | | | | | | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3COMADDG | Channel 3 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 30 | CH2COMADDG | Channel 2 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 29 | CH1COMADDG | Channel 1 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 28 | CH0COMADDG | Channel 0 additional compare event generation.<br>This bit is set by software to generate a compare event in channel 0 additional, it is automatically cleared by hardware.<br>When this bit is set, the CH0COMADDIF flag will be set, and the corresponding interrupt will be sent if enabled.<br>0: No generate a channel 0 additional compare event<br>1: Generate a channel 0 additional compare event<br>**Note:** This bit just used in composite PWM mode. |
| 27:24 | Reserved | Must be kept at reset value. |
| 23 | MCH3G | Multi mode channel 3 capture or compare event generation.<br>Refer to MCH0G description. |
| 22 | MCH2G | Multi mode channel 2 capture or compare event generation.<br>Refer to MCH0G description. |
| 21 | MCH1G | Multi mode channel 1 capture or compare event generation.<br>Refer to MCH0G description. |
| 20 | MCH0G | Multi mode channel 0 capture or compare event generation.<br>This bit is set by software to generate a capture or compare event in multi mode |

| | | |
|---|---|---|
| | | channel 0, it is automatically cleared by hardware. When this bit is set, the MCH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if multi mode channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_MCH0CV register, and the MCH0OF flag is set if the MCH0IF flag has been set.<br>0: No generate a multi mode channel 0 capture or compare event<br>1: Generate a multi mode channel 0 capture or compare event |
| 19:9 | Reserved | Must be kept at reset value. |
| 8 | BRK1G | BREAK1 event generation<br>This bit is set by software to generate an event and cleared by hardware automatically. When this bit is set, the POEN bit will be cleared and BRK1IF flag will be set.<br>0: No generate a BREAK1 event<br>1: Generate a BREAK1 event |
| 7 | BRK0G | BREAK0 event generation<br>This bit is set by software to generate an event and cleared by hardware automatically. When this bit is set, the POEN bit will be cleared and BRK0IF flag will be set.<br>0: No generate a BREAK0 event<br>1: Generate a BREAK0 event |
| 6 | TRGG | Trigger event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register will be set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a trigger event<br>1: Generate a trigger event |
| 5 | CMTG | Channel commutation event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, MCHxEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).<br>0: No affect<br>1: Generate channel commutation update event |
| 4 | CH3G | Channel 3 capture or compare event generation<br>Refer to CH0G description |
| 3 | CH2G | Channel 2 capture or compare event generation<br>Refer to CH0G description |
| 2 | CH1G | Channel 1 capture or compare event generation<br>Refer to CH0G description |
| 1 | CH0G | Channel 0 capture or compare event generation |

This bit is set by software to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag has been set.

0: No generate a channel 0 capture or compare event

1: Generate a channel 0 capture or compare event

| | | |
|---|---|---|
| 0 | UPG | Update event generation<br>This bit can be set by software, and automatically cleared by hardware. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, while in down counting mode it takes the auto-reload value. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1MS[2] | CH0MS[2] | CH1COMADDSEN | CH0COMADDSEN | Reserved | | | CH1COMCTL[3] | Reserved | | | | | | | CH0COMCTL[3] |
| | | Reserved | Reserved | | | | Reserved | | | | | | | | Reserved |
| rw | rw | rw | rw | | | | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1COMCEN | CH1COMCTL[2:0] | | | CH1COMSEN | Reserved | CH1MS[1:0] | | CH0COMCEN | CH0COMCTL[2:0] | | | CH0COMSEN | Reserved | CH0MS[1:0] | |
| CH1CAPFLT[3:0] | | | | CH1CAPPSC[1:0] | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH1MS[2] | Channel 1 I/O mode selection<br>Refer to CH1MS[1:0]description |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Refer to CH0MS[1:0] description |
| 29 | CH1COMADDSEN | Channel 1 additional compare output shadow enable<br>Refer to CH0COMADDSEN description. |
| 28 | CH0COMADDSEN | Channel 0 additional compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0COMV_ADD register which updates at each update event will be enabled. |

| | | |
|---|---|---|
| | | 0: Channel 0 additional compare output shadow disabled<br>1: Channel 0 additional compare output shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000. |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | CH1COMCTL[3] | Channel 1 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | CH0COMCTL[3] | Channel 0 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 15 | CH1COMCEN | Channel 1 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH1COMCTL[2:0] | Channel 1 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 11 | CH1COMSEN | Channel 1 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. The CH1MS[2:0] bit-field is writable only when the channel is not active (When MCH1MSEL[1:0] = 2b'00, the CH1EN bit in TIMERx_CHCTL2 register is reset; when MCH1MSEL[1:0] = 2b'11, the CH1EN and MCH1EN bits in TIMERx_CHCTL2 register are reset).<br>000: Channel 1 is configured as output.<br>001: Channel 1 is configured as input, IS1 is connected to CI1FE1.<br>010: Channel 1 is configured as input, IS1 is connected to CI0FE1.<br>011: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Channel 1 is configured as input, IS1 is connected to MCI1FE1.<br>101~111: Reserved. |
| 7 | CH0COMCEN | Channel 0 output compare clear enable<br>When this bit is set, the O0CPRE signal is cleared when high level is detected on ETIFP input.<br>0: Channel 0 output compare clear disabled<br>1: Channel 0 output compare clear enabled |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control |

The CH0COMCTL[3] and CH0COMCTL[2:0] bit-field control the behavior of O0CPRE which drives CH0_O. The active level of O0CPRE is high, while the active level of CH0_O depends on CH0P bit.

**Note**: When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, This bit-field controls the behavior of O0CPRE which drives CH0_O and MCH0_O. The active level of O0CPRE is high, while the active level of CH0_O and MCH0_O depends on CH0P and MCH0P bits.

0000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.

0001: Set the channel output on match. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.

0010: Clear the channel output on match. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.

0011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.

0100: Force low. O0CPRE is forced low level.

0101: Force high. O0CPRE is forced high level.

0110: PWM mode 0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV, otherwise it is inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV, otherwise it is active.

0111: PWM mode 1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV, otherwise it is active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV, otherwise it is inactive.

1000: Delayable SPM mode 0. The behavior of O0CPRE is performed as in PWM mode 0. When counting up, the O0CPRE is active. When a trigger event occurs, the O0CPRE is inactive. The O0CPRE is active again at the next update event; When counting down, the O0CPRE is inactive, when a trigger event occurs, the O0CPRE is active. The O0CPRE is inactive again at the next update event.

1001: Delayable SPM mode 1. The behavior of O0CPRE is performed as in PWM mode 1. When counting up, the O0CPRE is inactive, when a trigger event occurs, the O0CPRE is active. The O0CPRE is inactive again at the next update event; When counting down, the O0CPRE is active. When a trigger event occurs, the O0CPRE is inactive. The O0CPRE is active again at the next update event.

1010~1111: Reserved.

**Note**: In the composite PWM mode (CH0CPWMEN = 1'b1 and CH0MS = 3'b000), the PWM signal output in channel 0 is composited by TIMERx_CH0CV and TIMERx_CH0COMV_ADD. Please refer to ***Composite PWM mode*** for more details.

If configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from "Timing" mode to "PWM" mode or the result of the comparison changes.

When the outputs of CH0 and MCH0 are complementary, this bit-field is preloaded.

| | | |
|---|---|---|
| | | If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000 (compare mode). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register which updates at each update event will be enabled.<br>0: Channel 0 output compare shadow disabled<br>1: Channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. The CH0MS[2:0] bit-field is writable only when the channel is not active (When MCH0MSEL[1:0] = 2b'00, the CH1EN bit in TIMERx_CHCTL2 register is reset; when MCH0MSEL[1:0] = 2b'11, the CH0EN and MCH0EN bits in TIMERx_CHCTL2 register are reset).<br>000: Channel 0 is configured as output.<br>001: Channel 0 is configured as input, IS0 is connected to CI0FE0.<br>010: Channel 0 is configured as input, IS0 is connected to CI1FE0.<br>011: Channel 0 is configured as input, IS0 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Channel 0 is configured as input, IS0 is connected to MCI0FE0.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH1MS[2] | Channel 1 I/O mode selection<br>Same as output compare mode. |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Same as output compare mode. |
| 29:16 | Reserved | Must be kept at reset value. |
| 15:12 | CH1CAPFLT[3:0] | Channel 1 input capture filter control<br>Refer to CH0CAPFLT description. |
| 11:10 | CH1CAPPSC[1:0] | Channel 1 input capture prescaler<br>Refer to CH0CAPPSC description. |

| | | |
|---|---|---|
| 9:8 | CH1MS[1:0] | Channel 1 I/O mode selection<br>Same as output compare mode. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>Same as output compare mode. |

### Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3MS [2] | CH2MS [2] | CH3COM ADDSEN | CH2COM ADDSEN | Reserved | | | CH3COM CTL[3] | | | | | | | | CH2COM CTL[3] |
| | | Reserved | Reserved | | | | Reserved | | | | | | | | Reserved |
| rw | rw | rw | rw | | | | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMCEN | CH3COMCTL[2:0] | | | CH3COMSEN | Reserved | CH3MS[1:0] | | CH2COMCEN | CH2COMCTL[2:0] | | | CH2COMSEN | Reserved | CH2MS[1:0] | |
| CH3CAPFLT[3:0] | | | | CH3CAPPSC[1:0] | | | | CH2CAPFLT[3:0] | | | | CH2CAPPSC[1:0] | | | |
| rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3MS[2] | Channel 3 I/O mode selection<br>Refer to CH3MS[1:0]description. |
| 30 | CH2MS[2] | Channel 2 I/O mode selection<br>Refer to CH2MS[1:0] description. |
| 29 | CH3COMADDSEN | Channel 3 additional compare output shadow enable<br>Refer to CH2COMADDSEN description. |
| 28 | CH2COMADDSEN | Channel 2 additional compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH2COMV_ADD register which updates at each update event will be enabled.<br>0: Channel 2 additional compare shadow disabled<br>1: Channel 2 additional compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH2MS bit-field is 000. |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | CH3COMCTL[3] | Channel 3 compare output control<br>Refer to CH2COMCTL[2:0] description |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | CH2COMCTL[3] | Channel 2 compare output control<br>Refer to CH2COMCTL[2:0] description |
| 15 | CH3COMCEN | Channel 3 output compare clear enable<br>Refer to CH2COMCEN description |
| 14:12 | CH3COMCTL[2:0] | Channel 3 compare output control<br>Refer to CH2COMCTL[2:0] description |
| 11 | CH3COMSEN | Channel 3 output compare shadow enable<br>Refer to CH2COMSEN description |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CH3MS[1:0] | Channel 3 I/O mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. The CH3MS[2:0] bit-field is writable only when the channel is not active (When |

| | | |
|---|---|---|
| | | MCH3MSEL[1:0] = 2b'00, the CH3EN bit in TIMERx_CHCTL2 register is reset; when MCH3MSEL[1:0] = 2b'11, the CH3EN and MCH3EN bits in TIMERx_CHCTL2 register are reset).<br>00: Channel 3 is configured as output.<br>01: Channel 3 is configured as input, IS3 is connected to CI3FE3.<br>10: Channel 3 is configured as input, IS3 is connected to CI2FE3.<br>11: Channel 3 is configured as input, IS3 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Channel 3 is configured as input, IS3 is connected to MCI3FE3.<br>101~111: Reserved. |
| 7 | CH2COMCEN | Channel 2 output compare clear enable.<br>When this bit is set, the O2CPRE signal is cleared when high level is detected on ETIFP input.<br>0: Channel 2 output compare clear disabled<br>1: Channel 2 output compare clear enabled |
| 6:4 | CH2COMCTL[2:0] | Channel 2 compare output control<br>This bit-field controls the behavior of O2CPRE which drives CH2_O. The active level of O2CPRE is high, while the active level of CH2_O depends on CH2P bit.<br>**Note**: When multi mode channel 2 is configured in output mode, and the MCH2MSEL[1:0] = 2b'11, This bit-field controls the behavior of O2CPRE which drives CH2_O and MCH2_O. The active level of O2CPRE is high, while the active level of CH2_O and MCH2_O depends on CH2P and MCH2P bits.<br>0000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. O2CPRE signal is forced high when the counter matches the output compare register TIMERx_CH2CV.<br>0010: Clear the channel output on match. O2CPRE signal is forced low when the counter matches the output compare register TIMERx_CH2CV.<br>0011: Toggle on match. O2CPRE toggles when the counter matches the output compare register TIMERx_CH2CV.<br>0100: Force low. O2CPRE is forced low level.<br>0101: Force high. O2CPRE is forced high level.<br>0110: PWM mode 0. When counting up, O2CPRE is active as long as the counter is smaller than TIMERx_CH2CV, otherwise it is inactive. When counting down, O2CPRE is inactive as long as the counter is larger than TIMERx_CH2CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, O2CPRE is inactive as long as the counter is smaller than TIMERx_CH2CV, otherwise it is active. When counting down, O2CPRE is active as long as the counter is larger than TIMERx_CH2CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of O2CPRE is performed as in PWM |

mode 0. When counting up, the O2CPRE is active. When a trigger event occurs, the O2CPRE is inactive. The O2CPRE is active again at the next update event; When counting down, the O2CPRE is inactive, when a trigger event occurs, the O2CPRE is active. The O2CPRE is inactive again at the next update event.

1001: Delayable SPM mode 1. The behavior of O2CPRE is performed as in PWM mode 1. When counting up, the O2CPRE is inactive, when a trigger event occurs, the O2CPRE is active. The O2CPRE is inactive again at the next update event; When counting down, the O2CPRE is active. When a trigger event occurs, the O2CPRE is inactive. The O2CPRE is active again at the next update event.

1010~1111: Reserved.

**Note**: In the composite PWM mode (CH2CPWMEN = 1'b1 and CH2MS = 3'b000), the PWM signal output in channel 2 is composited by TIMERx_CH2CV and TIMERx_CH2COMV_ADD. Please refer to ***Composite PWM mode*** for more details.

If configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from "Timing" mode to "PWM" mode or the result of the comparison changes.

When the outputs of CH2 and MCH2 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.

This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH2MS bit-field is 000(compare mode).

| | | |
|---|---|---|
| 3 | CH2COMSEN | Channel 2 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled.<br>0: Channel 2 output compare shadow disabled<br>1: Channel 2 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH2MS bit-field is 000. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | CH2MS[1:0] | Channel 2 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. The CH2MS[2:0] bit-field is writable only when the channel is not active (When MCH2MSEL[1:0] = 2b'00, the CH2EN bit in TIMERx_CHCTL2 register is reset; when MCH2MSEL[1:0] = 2b'11, the CH2EN and MCH2EN bits in TIMERx_CHCTL2 register are reset).<br>00: Channel 2 is configured as output.<br>01: Channel 2 is configured as input, IS2 is connected to CI2FE2.<br>10: Channel 2 is configured as input, IS2 is connected to CI3FE2.<br>11: Channel 2 is configured as input, IS2 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in |

| | | |
|---|---|---|
| | | SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Channel 2 is configured as input, IS2 is connected to MCI2FE2.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3MS[2] | Channel 3 I/O mode selection<br>Same as output compare mode. |
| 30 | CH2MS[2] | Channel 2 I/O mode selection<br>Same as output compare mode. |
| 31:16 | Reserved | Must be kept at reset value. |
| 15:12 | CH3CAPFLT[3:0] | Channel 3 input capture filter control<br>Refer to CH2CAPFLT description. |
| 11:10 | CH3CAPPSC[1:0] | Channel 3 input capture prescaler<br>Refer to CH2CAPPSC description. |
| 9:8 | CH3MS[1:0] | Channel 3 mode selection<br>Same as output compare mode. |
| 7:4 | CH2CAPFLT[3:0] | Channel 2 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI2 input signal and the length of the digital filter applied to CI2.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | CH2CAPPSC[1:0] | Channel 2 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 register is cleared. |

| | | |
|---|---|---|
| | | 00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | CH2MS[1:0] | Channel 2 mode selection<br>Same as output compare mode. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MCH3P | MCH3EN | CH3P | CH3EN | MCH2P | MCH2EN | CH2P | CH2EN | MCH1P | MCH1EN | CH1P | CH1EN | MCH0P | MCH0EN | CH0P | CH0EN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | MCH3P | Multi mode channel 3 capture/compare polarity<br>Refer to MCH0P description. |
| 14 | MCH3EN | Multi mode channel 3 capture/compare enable<br>Refer to MCH0EN description. |
| 13 | CH3P | Channel 3 capture/compare polarity<br>Refer to CH0P description. |
| 12 | CH3EN | Channel 3 capture/compare enable<br>Refer to CH0EN description. |
| 11 | MCH2P | Multi mode channel 2 output polarity<br>Refer to MCH0P description. |
| 10 | MCH2EN | Multi mode channel 2 output enable<br>Refer to MCH0EN description. |
| 9 | CH2P | Channel 2 capture/compare polarity<br>Refer to CH0P description. |
| 8 | CH2EN | Channel 2 capture/compare enable<br>Refer to CH0EN description. |
| 7 | MCH1P | Multi mode channel 1 output polarity |

| | | |
|---|---|---|
| | | Refer to MCH0P description. |
| 6 | MCH1EN | Multi mode channel 1 output enable<br>Refer to MCH0EN description. |
| 5 | CH1P | Channel 1 capture/compare polarity<br>Refer to CH0P description. |
| 4 | CH1EN | Channel 1 capture/compare enable<br>Refer to CH0EN description. |
| 3 | MCH0P | Multi mode channel 0 output polarity<br>When Multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, this bit specifies the MCH0_O output signal polarity.<br>0: Multi mode channel 0 output active high<br>1: Multi mode channel 0 output active low<br>When CH0 is configured in input mode, in conjunction with CH0P, this bit is used to define the polarity of CH0.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 2 | MCH0EN | Multi mode channel 0 capture/compare enable<br>When multi mode channel 0 is configured in output mode, setting this bit enables MCH0_O signal in active state. When multi mode channel 0 is configured in input mode, setting this bit enables the capture event in multi mode channel 0.<br>0: Multi mode channel 0 disabled<br>1: Multi mode channel 0 enabled |
| 1 | CH0P | Channel 0 capture/compare polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, these bits specifie the channel 0 input signal's polarity. [MCH0P, CH0P] will select the active trigger or capture polarity for channel 0 input signals.<br>00: channel 0 input signal's rising edge is the active signal for capture or trigger operation in slave mode. And channel 0 input signal will not be inverted.<br>01: channel 0 input signal's falling edge is the active signal for capture or trigger operation in slave mode. And channel 0 input signal will be inverted.<br>10: Reserved.<br>11: Noninverted/ both channel 0 input signal's edges.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables |

the capture event in channel 0.

0: Channel 0 disabled

1: Channel 0 enabled

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UPIFBU | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:0 | CNT[15:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

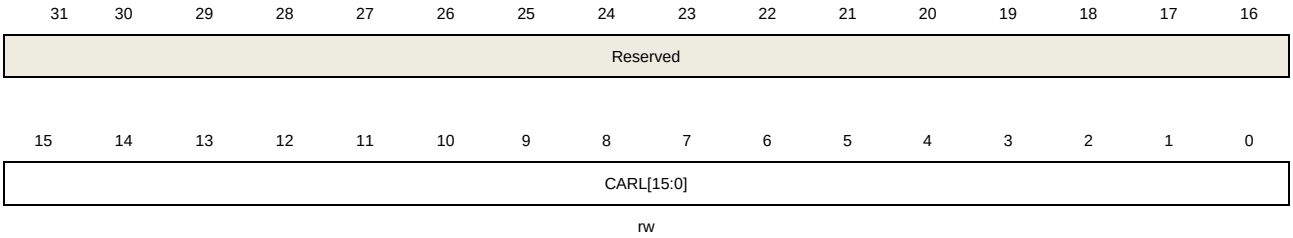
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of |

this bit-field will be loaded to the corresponding shadow register at every update event.

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 FFFF

This register has to be accessed by word (32-bit).

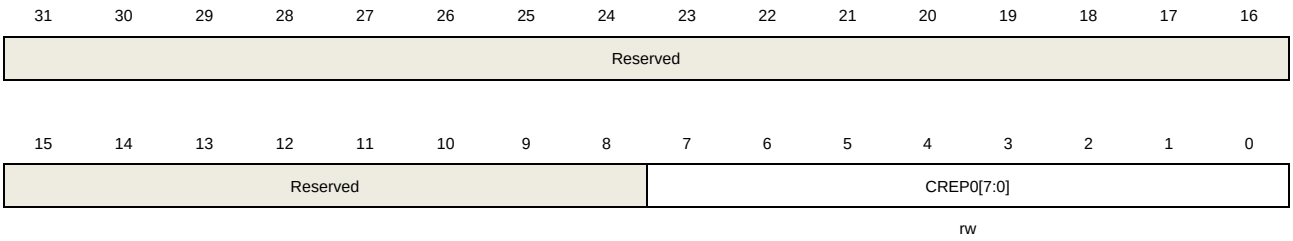



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-field specifies the auto reload value of the counter. |

### Counter repetition register 0 (TIMERx_CREP0)

Address offset: 0x30
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



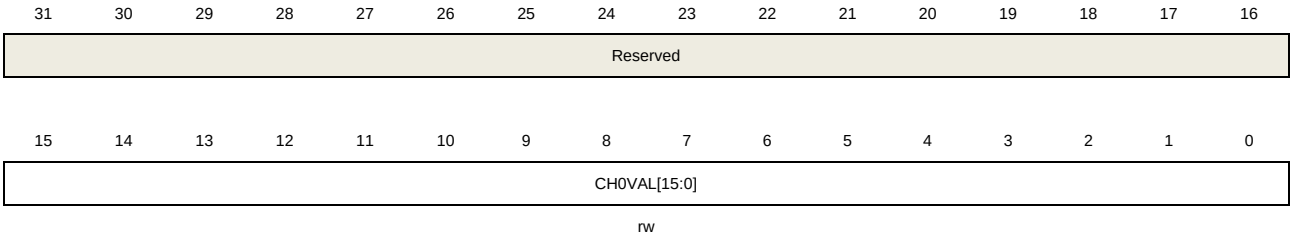

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | CREP0[7:0] | Counter repetition value 0<br>This bit-field specifies the update event generation rate. Each time the repetition counter counts down to zero, an update event will be generated. The update rate of the shadow registers is also affected by this bit-field when these shadow registers are enabled.<br>**Note**: This bit-field just used with CREPSEL =0 (in TIMERx_CFG register). |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



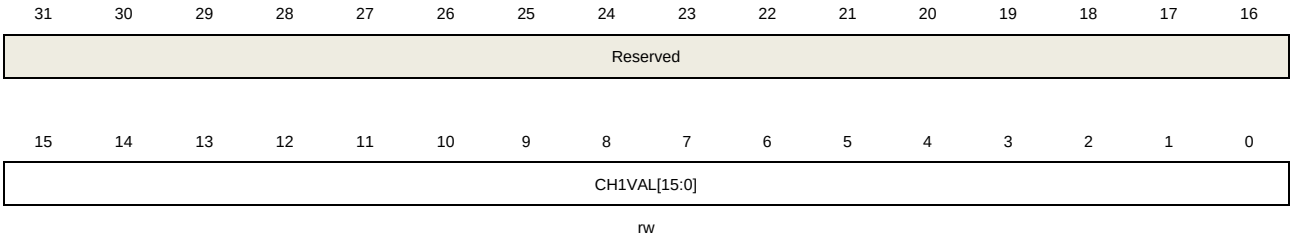

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH0VAL[15:0] | Capture/compare value of channel 0<br>When channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



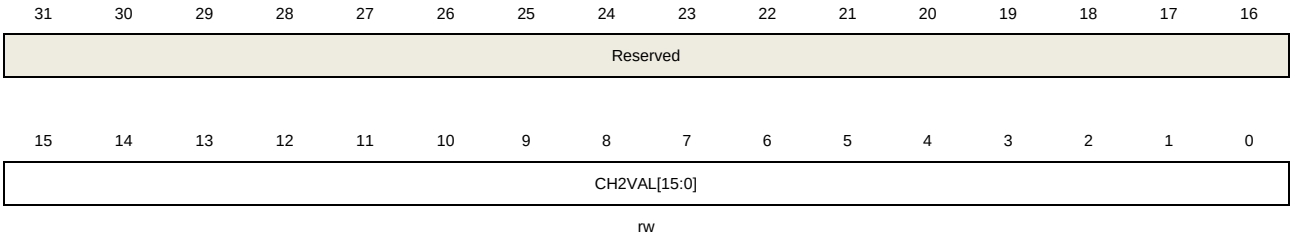

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH1VAL[15:0] | Capture/compare value of channel 1<br>When channel 1 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 2 capture/compare value register (TIMERx_CH2CV)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



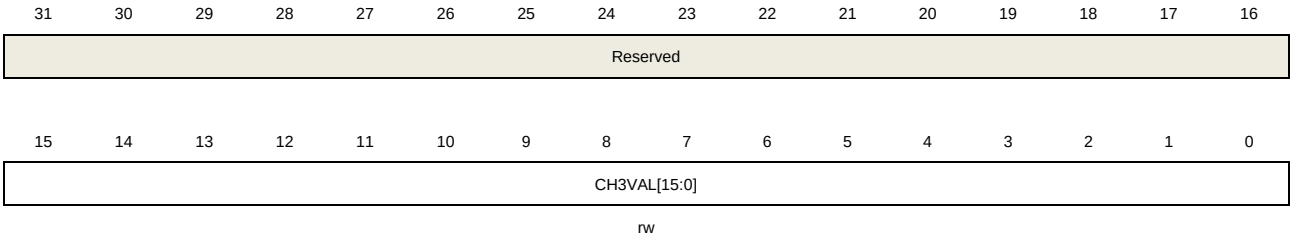

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH2VAL[15:0] | Capture/compare value of channel 2<br>When channel 2 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 2 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 3 capture/compare value register (TIMERx_CH3CV)

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH3VAL[15:0] | Capture/compare value of channel 3<br>When channel 3 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | BRK1LK | BRK0LK | BRK1REL | BRK0REL | BRK1P | BRK1EN | BRK1F[3:0] | | | | BRK0F[3:0] | | | |
| | | rw | rw | rw | rw | rw | rw | rw | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POEN | OAEN | BRK0P | BRK0EN | ROS | IOS | PROT[1:0] | | DTCFG[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | BRK1LK | BREAK1 input locked<br>Refer to BRK0LK description |
| 28 | BRK0LK | BREAK0 input locked<br>0: BREAK0 input in input mode<br>1: BREAK0 input in locked mode<br>When the BRK0LK is set to 1, the BREAK0 input is configured in open drain output mode.<br>Any active BREAK0 event asserts a low logic level on the BREAK0 input to indicate an internal BREAK0 event to external devices.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.<br>**Note**: Every write operation to this bit needs a delay of 1 APB clock to active. |
| 27 | BRK1REL | BREAK1 input released<br>Refer to BRK0REL description. |
| 26 | BRK0REL | BREAK0 input released<br>This bit is cleared by hardware when the BREAK0 input is invalid.<br>0: BREAK0 input is unreleased<br>1: BREAK0 input is released<br>The locked output control (open drain mode in Hi-z state) is released by setting this bit with software. And when the fault is disappeared, this bit will reset by hardware.<br>**Note**: Every write operation to this bit needs a delay of 1 APB clock to active. |
| 25 | BRK1P | BREAK1 input signal polarity<br>This bit specifies the polarity of the BREAK1 input signal.<br>0: BREAK1 input active low<br>1: BREAK1 input active high<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

**Note**: Every write operation to this bit needs a delay of 1 APB clock to active.

24 BRK1EN

BREAK1 input signal enable

This bit can be set to enable the BREAK1 input signal.

0: BREAK1 input disabled

1: BREAK1 input enabled

This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

**Note**:

1) Every write operation to this bit needs a delay of 1 APB clock to active.
2) This bit is only used with ROS=1 and IOS=1.

23:20 BRK1F[3:0]

BREAK1 input signal filter

An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample BREAK1 input signal and the length of the digital filter applied to BREAK1.

0000: Filter disabled. BREAK1 act asynchronously, N=1

0001: $f_{SAMP} = f_{CK_TIMER}$, N=2

0010: $f_{SAMP} = f_{CK_TIMER}$, N=4

0011: $f_{SAMP} = f_{CK_TIMER}$, N=8

0100: $f_{SAMP} = f_{DTS}/2$, N=6

0101: $f_{SAMP} = f_{DTS}/2$, N=8

0110: $f_{SAMP} = f_{DTS}/4$, N=6

0111: $f_{SAMP} = f_{DTS}/4$, N=8

1000: $f_{SAMP} = f_{DTS}/8$, N=6

1001: $f_{SAMP} = f_{DTS}/8$, N=8

1010: $f_{SAMP} = f_{DTS}/16$, N=5

1011: $f_{SAMP} = f_{DTS}/16$, N=6

1100: $f_{SAMP} = f_{DTS}/16$, N=8

1101: $f_{SAMP} = f_{DTS}/32$, N=5

1110: $f_{SAMP} = f_{DTS}/32$, N=6

1111: $f_{SAMP} = f_{DTS}/32$, N=8

This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

19:16 BRK0F[3:0]

BREAK0 input signal filter

An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample BREAK0 input signal and the length of the digital filter applied to BREAK0.

0000: Filter disabled. BREAK0 act asynchronously, N=1

0001: $f_{SAMP} = f_{CK_TIMER}$, N=2

0010: $f_{SAMP} = f_{CK_TIMER}$, N=4

0011: $f_{SAMP} = f_{CK_TIMER}$, N=8

0100: $f_{SAMP} = f_{DTS}/2$, N=6

0101: $f_{SAMP} = f_{DTS}/2$, N=8

0110: $f_{SAMP} = f_{DTS}/4$, N=6

0111: $f_{SAMP} = f_{DTS}/4$, N=8
1000: $f_{SAMP} = f_{DTS}/8$, N=6
1001: $f_{SAMP} = f_{DTS}/8$, N=8
1010: $f_{SAMP} = f_{DTS}/16$, N=5
1011: $f_{SAMP} = f_{DTS}/16$, N=6
1100: $f_{SAMP} = f_{DTS}/16$, N=8
1101: $f_{SAMP} = f_{DTS}/32$, N=5
1110: $f_{SAMP} = f_{DTS}/32$, N=6
1111: $f_{SAMP} = f_{DTS}/32$, N=8
This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

| | | |
|---|---|---|
| 15 | POEN | Primary output enable<br>This bit is set by software or automatically set by hardware depending on the OAEN bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and MCHx_O) if the corresponding enable bits (CHxEN, MCHxEN in TIMERx_CHCTL2 register) have been set.<br>0: Channel outputs are disabled or forced to idle state.<br>1: Channel outputs are enabled. |
| 14 | OAEN | Output automatic enable<br>This bit specifies whether the POEN bit can be set automatically by hardware.<br>0: POEN cannot be set by hardware.<br>1: POEN can be set by hardware automatically at the next update event, if the break input is not active.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 13 | BRK0P | BREAK0 input signal polarity<br>This bit specifies the polarity of the BREAK0 input signal.<br>0: BREAK0 input active low<br>1: BREAK0 input active high<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 12 | BRK0EN | BREAK0 input signal enable<br>This bit can be set to enable the BREAK0 input signal<br>0: BREAK0 input disabled<br>1: BREAK0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 24-4. Complementary outputs controlled by parameters (MCHxMSEL =2’b11)***. |

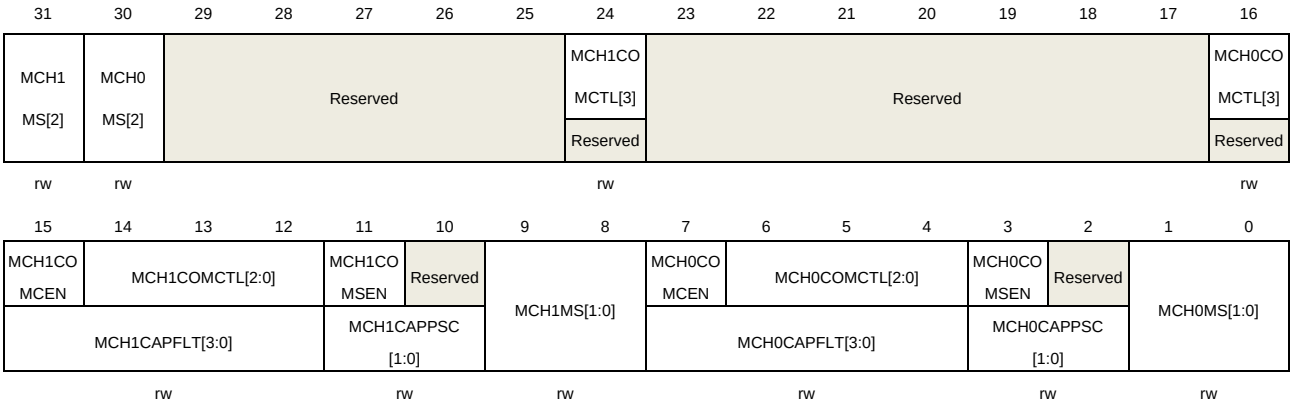
| | | |
|---|---|---|
| | | 0: “off-state” disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.<br>1: “off-state” enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 24-4. Complementary outputs controlled by parameters (MCHxMSEL =2’b11)***.<br>0: “off-state” disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CHxEN/CHxNEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 9:8 | PROT[1:0] | Complementary register protect control<br>This bit-field specifies the write protection property of registers.<br>00: Protect disabled. No write protection.<br>01: PROT mode 0. The ISOx/ISOxN bits in TIMERx_CTL1 register, the BRK0EN/ BRK0P/BRK1EN/ BRK1P/OAEN/DTCFG bits in TIMERx_CCHP register, the DTCFG bits in TIMERx_FCCHPx (x = 0...3) register, are writing protected.<br>10: PROT mode 1. In addition to the registers in PROT mode 0, the CHxP/MCHxP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode), the ROS/IOS bits in TIMERx_CCHP register and the ROS/IOS bits in TIMERx_FCCHPx (x = 0..3) register are writing protected.<br>11: PROT mode 2. In addition to the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN/ CHxCOMADDSEN/ MCHxCOMCTL/ MCHxCOMSEN bits in TIMERx_CHCTL0/1 and TIMERx_MCHCTL0/1 registers (if the related channel is configured in output) are writing protected.<br>This bit-field can be written only once after the system reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected. |
| 7:0 | DTCFG[7:0] | Dead time configuration<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between the value of DTCFG and the duration of dead-time is as follow:<br>DTCFG[7:5] = 3’b0xx: DT value = DTCFG[7:0] * $t_{DT}$, $t_{DT}$ = $t_{DTS}$.<br>DTCFG[7:5] = 3’b10x: DT value = (64+DTCFG[5:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*2.<br>DTCFG[7:5] = 3’b110: DT value = (32+DTCFG[4:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*8.<br>DTCFG[7:5] = 3’b111: DT value = (32+DTCFG[4:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*16.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### Multi mode channel control register 0 (TIMERx_MCHCTL0)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MCH1MS[2] | Multi mode channel 1 I/O mode selection<br>Refer to MCH1MS[1:0]description. |
| 30 | MCH0MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH0MS[1:0] description. |
| 29:25 | Reserved | Must be kept at reset value. |
| 24 | MCH1COMCTL[3] | Multi mode channel 1 compare output control.<br>Refer to MCH0COMCTL[2:0] description. |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | MCH0COMCTL[3] | Multi mode channel 0 compare output control.<br>Refer to MCH0COMCTL[2:0] description. |
| 15 | MCH1COMCEN | Multi mode channel 1 output compare clear enable.<br>Refer to MCH0COMCEN description. |
| 14:12 | MCH1COMCTL[2:0] | Multi mode channel 1 compare output control.<br>Refer to MCH0COMCTL description. |
| 11 | MCH1COMSEN | Multi mode channel 1 output compare shadow enable<br>Refer to MCH0COMSEN description. |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | MCH1MS[1:0] | Multi mode channel 1 I/O mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active (the MCH1EN bit in TIMERx_CHCTL2 register is reset) |

| | | |
|---|---|---|
| | | 000: Multi mode channel 1 is configured as output.<br>001: Multi mode channel 1 is configured as input, MIS1 is connected to MCI1FEM1.<br>010: Multi mode channel 1 is configured as input, MIS1 is connected to MCI0FEM1.<br>011: Multi mode channel 1 is configured as input, MIS1 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Multi mode channel 1 is configured as input, MIS1 is connected to CI1FEM1.<br>101~111: Reserved. |
| 7 | MCH0COMCEN | Multi mode channel 0 output compare clear enable.<br>When this bit is set, the MO0CPRE signal is cleared when high level is detected on ETIFP input.<br>0: Multi mode channel 0 output compare clear disabled.<br>1: Multi mode channel 0 output compare clear enabled. |
| 6:4 | MCH0COMCTL [2:0] | Multi mode channel 0 output compare control<br>When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'00, the MCH0COMCTL[3] and MCH0COMCTL[2:0] bit-field control the behavior of MO0CPRE which drives MCH0_O. The active level of MO0CPRE is high, while the active level of MCH0_O depends on MCH0FP[1:0] bits.<br>**Note**: When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, the CH0COMCTL[2:0] bit-field controls the behavior of O0CPRE which drives CH0_O and MCH0_O, while the active level of CH0_O and MCH0_O depends on CH0P and MCH0P bits.<br>0000: Timing mode. The MO0CPRE signal keeps stable, independent of the comparison between the register TIMERx_MCH0CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. MO0CPRE signal is forced high when the counter matches the output compare register TIMERx_MCH0CV.<br>0010: Clear the channel output on match. MO0CPRE signal is forced low when the counter matches the output compare register TIMERx_MCH0CV.<br>0011: Toggle on match. MO0CPRE toggles when the counter matches the output compare register TIMERx_MCH0CV.<br>0100: Force low. MO0CPRE is forced low level.<br>0101: Force high. MO0CPRE is forced high level.<br>0110: PWM mode 0. When counting up, MO0CPRE is active as long as the counter is smaller than TIMERx_MCH0CV, otherwise it is inactive. When counting down, MO0CPRE is inactive as long as the counter is larger than TIMERx_MCH0CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, MO0CPRE is inactive as long as the counter is smaller than TIMERx_MCH0CV, otherwise it is active. When counting down, MO0CPRE is active as long as the counter is larger than TIMERx_MCH0CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of MO0CPRE is performed as in PWM mode 0. When counting up, the MO0CPRE is active. When a trigger event occurs, |

| | | |
|---|---|---|
| | | the MO0CPRE is inactive. The MO0CPRE is active again at the next update event; When counting down, the MO0CPRE is inactive, when a trigger event occurs, the O0CPR MO0CPRE E is active. The MO0CPRE is inactive again at the next update event.<br>1001: Delayable SPM mode 1. The behavior of MO0CPRE is performed as in PWM mode 1. When counting up, the MO0CPRE is inactive, when a trigger event occurs, the MO0CPRE is active. The MO0CPRE is inactive again at the next update event; When counting down, the MO0CPRE is active. When a trigger event occurs, the MO0CPRE is inactive. The MO0CPRE is active again at the next update event.<br>1010~1111: Reserved.<br>If configured in PWM mode, the MO0CPRE level changes only when the output compare mode switches from “Timing” mode to “PWM” mode or the result of the comparison changes.<br>When the outputs of CH0 and MCH0 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0NMS bit-field is 00(compare mode). |
| 3 | MCH0COMSEN | Multi mode channel 0 output compare shadow enable<br>When this bit is set, the shadow register of TIMERx_MCH0CV register which updates at each update event will be enabled.<br>0: Multi mode channel 0 output compare shadow disabled<br>1: Multi mode channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and MCH0MS bit-field is 00. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | MCH0MS[1:0] | Multi mode channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active (MCH0EN bit in TIMERx_CHCTL2 register is reset).<br>000: Multi mode channel 0 is configured as output.<br>001: Multi mode channel 0 is configured as input, MIS0 is connected to MCI0FEM0.<br>010: Multi mode channel 0 is configured as input, MIS0 is connected to MCI1FEM0.<br>011: Multi mode channel 0 is configured as input, MIS0 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Multi mode channel 0 is configured as input, MIS0 is connected to CI0FEM0.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|

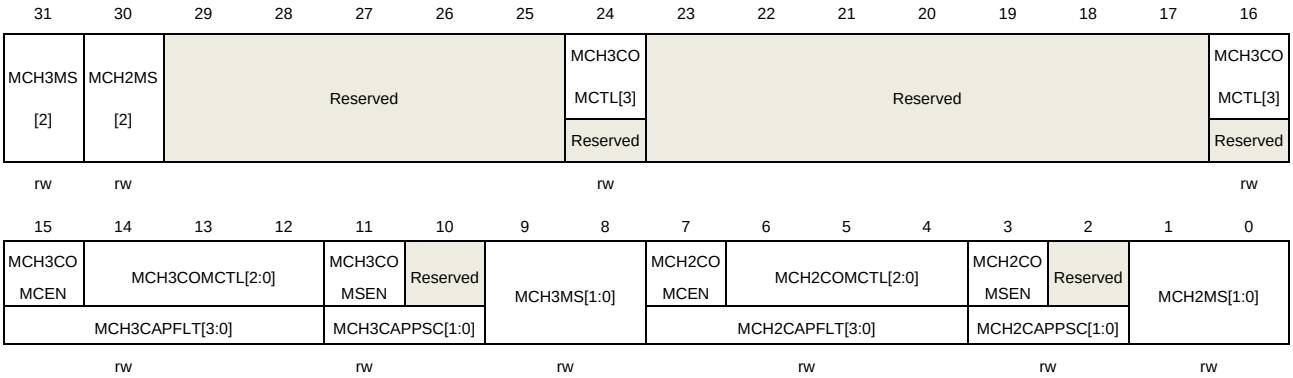
| | | |
|---|---|---|
| 31 | MCH1MS[2] | Multi mode channel 1 I/O mode selection<br>Refer to MCH1MS[1:0]description. |
| 30 | MCH0MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH0MS[1:0] description. |
| 29:16 | Reserved | Must be kept at reset value. |
| 15:12 | MCH1CAPFLT[3:0] | Multi mode channel 1 input capture filter control.<br>Refer to MCH0CAPFLT description. |
| 11:10 | MCH1CAPPSC[1:0] | Multi mode channel 1 input capture prescaler.<br>Refer to MCH0CAPPSC description. |
| 9:8 | MCH1MS[1:0] | Multi mode channel 1 I/O mode selection.<br>Same as output compare mode. |
| 7:4 | MCH0CAPFLT[3:0] | Multi mode channel 0 input capture filter control.<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample MCI0 input signal and the length of the digital filter applied to MCI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | MCH0CAPPSC[1:0] | Multi mode channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when MCH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | MCH0MS[1:0] | Multi mode channel 0 I/O mode selection<br>Same as output compare mode |

### Multi mode channel control register 1 (TIMERx_MCHCTL1)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MCH3MS[2] | Multi mode channel 1 I/O mode selection<br>Refer to MCH3MS[1:0]description. |
| 30 | MCH2MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH2MS[1:0] description. |
| 29:25 | Reserved | Must be kept at reset value. |
| 24 | MCH3COMCTL[3] | Multi mode channel 3 compare output control.<br>Refer to MCH2COMCTL[2:0] description. |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | MCH2COMCTL[3] | Multi mode channel 2 compare output control.<br>Refer to MCH2COMCTL[2:0] description. |
| 15 | MCH3COMCEN | Multi mode channel 3 output compare clear enable<br>Refer to MCH2COMCEN description |
| 14:12 | MCH3COMCTL[2:0] | Multi mode channel 3 compare output control<br>Refer to MCH2COMCTL description |
| 11 | MCH3COMSEN | Multi mode channel 3 output compare shadow enable<br>Refer to MCH2COMSEN description |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | MCH3MS[1:0] | Multi mode channel 3 I/O mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active (MCH3EN bit in TIMERx_CHCTL2 register is reset).<br>000: Multi mode channel 3 is configured as output. |

| | | |
|---|---|---|
| | | 01: Multi mode channel 3 is configured as input, MIS3 is connected to MCI3FEM3.<br>010: Multi mode channel 3 is configured as input, MIS3 is connected to MCI2FEM3.<br>011: Multi mode channel 3 is configured as input, MIS3 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Multi mode channel 3 is configured as input, MIS3 is connected to CI3FEM3.<br>101~111: Reserved. |
| 7 | MCH2COMCEN | Multi mode channel 2 output compare clear enable.<br>When this bit is set, the MO2CPRE signal is cleared when high level is detected on ETIFP input.<br>0: Multi mode channel 2 output compare clear disabled<br>1: Multi mode channel 2 output compare clear enabled |
| 6:4 | MCH2COMCTL[2:0] | Multi mode channel 2 compare output control<br>When multi mode channel 2 is configured in output mode, and the MCH2MSEL[1:0] = 2b'00, MCH2COMCTL[3] and MCH2COMCTL[2:0] bit-field control the behavior of MO2CPRE which drives MCH2_O. The active level of MO2CPRE is high, while the active level of MCH2_O depends on MCH2FP[1:0] bits.<br>**Note**: When multi mode channel 2 is configured in output mode, and the MCH2MSEL[1:0] = 2b'11, the CH2COMCTL[2:0] bit-field controls the behavior of O2CPRE which drives CH2_O and MCH2_O, while the active level of CH2_O and MCH2_O depends on CH2P and MCH2P bits.<br>0000: Timing mode. The MO2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_MCH2CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. MO2CPRE signal is forced high when the counter matches the output compare register TIMERx_MCH2CV.<br>0010: Clear the channel output on match. MO2CPRE signal is forced low when the counter matches the output compare register TIMERx_MCH2CV.<br>0011: Toggle on match. MO2CPRE toggles when the counter matches the output compare register TIMERx_MCH2CV.<br>0100: Force low. MO2CPRE is forced low level.<br>0101: Force high. MO2CPRE is forced high level.<br>0110: PWM mode 0. When counting up, MO2CPRE is active as long as the counter is smaller than TIMERx_MCH2CV, otherwise it is inactive. When counting down, MO2CPRE is inactive as long as the counter is larger than TIMERx_MCH2CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, MO2CPRE is inactive as long as the counter is smaller than TIMERx_MCH2CV, otherwise it is active. When counting down, MO2CPRE is active as long as the counter is larger than TIMERx_MCH2CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of MO2CPRE is performed as in PWM mode 0. When counting up, the MO2CPRE is active. When a trigger event occurs, the MO2CPRE is inactive. The MO2CPRE is active again at the next update event; |

| | | |
|---|---|---|
| | | When counting down, the MO2CPRE is inactive, when a trigger event occurs, the O0CPR MO2CPRE E is active. The MO2CPRE is inactive again at the next update event.<br>1001: Delayable SPM mode 1. The behavior of MO2CPRE is performed as in PWM mode 1. When counting up, the MO2CPRE is inactive, when a trigger event occurs, the MO2CPRE is active. The MO2CPRE is inactive again at the next update event; When counting down, the MO2CPRE is active. When a trigger event occurs, the MO2CPRE is inactive. The MO2CPRE is active again at the next update event.<br>1010~1111: Reserved.<br>If configured in PWM mode, the MO2CPRE level changes only when the output compare mode switches from “Timing” mode to “PWM” mode or the result of the comparison changes.<br>When the outputs of CH2 and MCH2 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH2NMS bit-field is 00(compare mode). |
| 3 | MCH2COMSEN | Multi mode channel 2 output compare shadow enable<br>When this bit is set, the shadow register of TIMERx_MCH2CV register, which updates at each update event will be enabled.<br>0: Multi mode channel 2 output compare shadow disabled<br>1: Multi mode channel 2 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH2NMS bit-field is 00. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | MCH2MS[1:0] | Multi mode channel 2 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active (MCH2EN bit in TIMERx_CHCTL2 register is reset).<br>000: Multi mode channel 2 is configured as output.<br>001: Multi mode channel 2 is configured as input, MIS2 is connected to MCI2FEM2.<br>010: Multi mode channel 2 is configured as input, MIS2 is connected to MCI3FEM2.<br>011: Multi mode channel 2 is configured as input, MIS2 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=0,7) register).<br>100: Multi mode channel 2 is configured as input, MIS2 is connected to CI2FEM2.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MCH3MS[2] | Multi mode channel 1 I/O mode selection |

| | | |
|---|---|---|
| | | Refer to MCH3MS[1:0]description. |
| 30 | MCH2MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH2MS[1:0] description. |
| 29:16 | Reserved | Must be kept at reset value. |
| 15:12 | MCH3CAPFLT[3:0] | Multi mode channel 3 input capture filter control.<br>Refer to MCH2CAPFLT description. |
| 11:10 | MCH3CAPPSC[1:0] | Multi mode channel 3 input capture prescaler.<br>Refer to MCH2CAPPSC description. |
| 9:8 | MCH3MS[1:0] | Multi mode channel 3 I/O mode selection.<br>Same as output compare mode. |
| 7:4 | MCH2CAPFLT[3:0] | Multi mode channel 2 input capture filter control.<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample MCI2 input signal and the length of the digital filter applied to MCI2.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | MCH2CAPPSC[1:0] | Multi mode channel 2 input capture prescaler.<br>This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when MCH2EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | MCH2MS[1:0] | Multi mode channel 2 I/O mode selection<br>Same as output compare mode. |

### Multi mode channel control register 2 (TIMERx_MCHCTL2)

Address offset: 0x50

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | MCH3FP[1:0] | | MCH2FP[1:0] | | MCH1FP[1:0] | | MCH0FP[1:0] | |
| | | | | | | | | rw | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:6 | MCH3FP[1:0] | Multi mode channel 3 capture/compare free polarity<br>Refer to MCH0FP[1:0] description. |
| 5:4 | MCH2FP[1:0] | Multi mode channel 2 capture/compare free polarity<br>Refer to MCH0FP[1:0] description. |
| 3:2 | MCH1FP[1:0] | Multi mode channel 1 capture/compare free polarity<br>Refer to MCH0FP[1:0] description. |
| 1:0 | MCH0FP[1:0] | Multi mode channel 0 capture/compare free polarity<br>When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'00, these bits specifie the multi mode channel 0 output signal polarity.<br>00: Multi mode channel 0 active high<br>01: Multi mode channel 0 active low<br>10: Reserved.<br>11: Reserved.<br>When multi mode channel 0 is configured in input mode, these bits specifie the multi mode channel 0 input signal's polarity. MCH0FP[1:0] will select the active trigger or capture polarity for multi mode channel 0 input signals.<br>00: Multi mode channel 0 input signal's rising edge is the active signal for capture or trigger operation in slave mode. And multi mode channel 0 input signal will not be inverted.<br>01: Multi mode channel 0 input signal's falling edge is the active signal for capture or trigger operation in slave mode. And multi mode channel 0 input signal will be inverted.<br>10: Reserved.<br>11: Noninverted/both multi mode channel 0 input signal's edges.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |

### Multi mode channel 0 capture/compare value register (TIMERx_MCH0CV)

Address offset: 0x54

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MCH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | MCH0VAL[15:0] | Capture/compare value of multi mode channel 0.<br>When multi mode channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When multi mode channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Multi mode channel 1 capture/compare value register (TIMERx_MCH1CV)

Address offset: 0x58

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MCH1VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

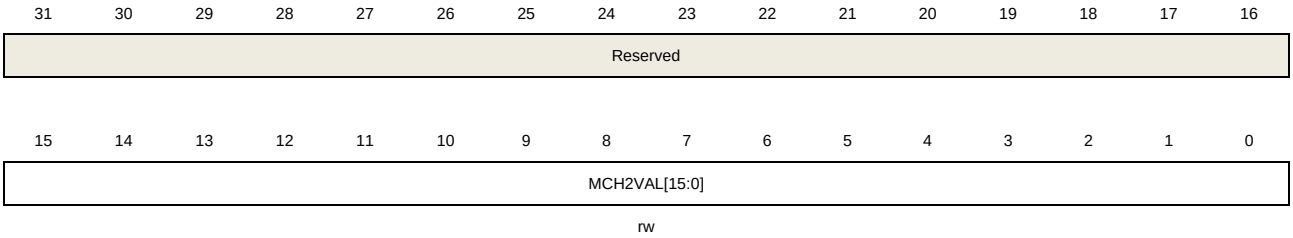
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | MCH1VAL[15:0] | Capture/compare value of multi mode channel 1.<br>When multi mode channel 1 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When multi mode channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Multi mode channel 2 capture/compare value register (TIMERx_MCH2CV)

Address offset: 0x5C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



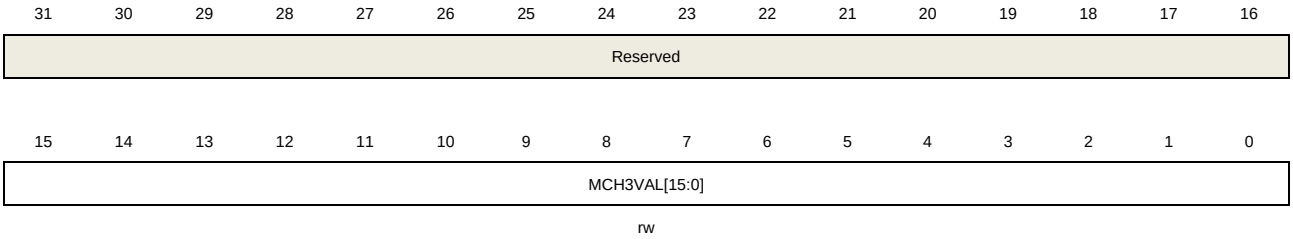

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | MCH2VAL[15:0] | Capture/compare value of multi mode channel 2.<br>When multi mode channel 2 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When multi mode channel 2 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Multi mode channel 3 capture/compare value register (TIMERx_MCH3CV)

Address offset: 0x60

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | MCH3VAL[15:0] | Capture/compare value of channel 3.<br>When multi mode channel 3 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When multi mode channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 0 additional compare value register (TIMERx_CH0COMV_ADD)

Address offset: 0x64

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



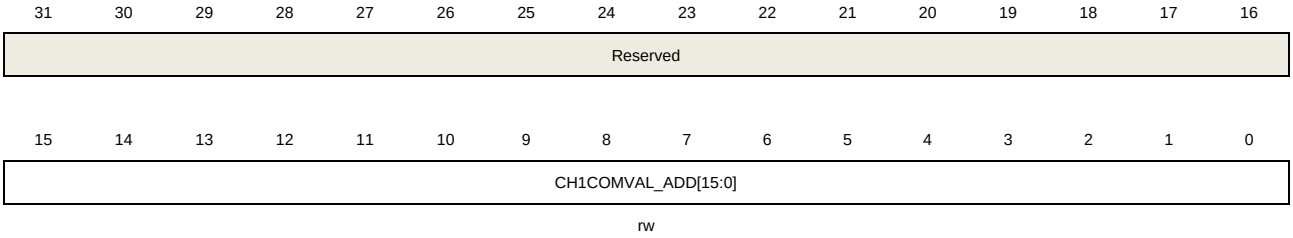

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH0COMVAL_ADD [15:0] | Additional compare value of channel 0<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 1 additional compare value register (TIMERx_CH1COMV_ADD)

Address offset: 0x68

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH1COMVAL_ADD [15:0] | Additional compare value of channel 1<br>When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 2 additional compare value register (TIMERx_CH2COMV_ADD)

Address offset: 0x6C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH2COMVAL_ADD [15:0] | Additional compare value of channel 2<br>When channel 2 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 3 additional compare value register (TIMERx_CH3COMV_ADD)

Address offset: 0x70

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH3COMVAL_ADD [15:0] | Additional compare value of channel 3<br>When channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Control register 2 (TIMERx_CTL2)

Address offset: 0x74

Reset value: 0x0FF0 00FF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3CPWMEN | CH2CPWMEN | CH1CPWMEN | CH0CPWMEN | MCH3MSEL[1:0] | | MCH2MSEL[1:0] | | MCH1MSEL[1:0] | | MCH0MSEL[1:0] | | DECDISDEN | DECJDEN | Reserved | |
| rw | rw | rw | rw | rw | | rw | | rw | | rw | | rw | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CH3OMPSEL[1:0] | | CH2OMPSEL[1:0] | | CH1OMPSEL[1:0] | | CH0OMPSEL[1:0] | | BRKENCH3 | BRKENCH2 | BRKENCH1 | BRKENCH0 | DTIENCH3 | DTIENCH2 | DTIENCH1 | DTIENCH0 |
| rw | | rw | | rw | | rw | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3CPWMEN | Channel 3 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 30 | CH2CPWMEN | Channel 2 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 29 | CH1CPWMEN | Channel 1 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 28 | CH0CPWMEN | Channel 0 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 27:26 | MCH3MSEL[1:0] | Multi mode channel 3 mode select<br>00: Independent mode, MCH3 is independent of CH3<br>01: Reserved<br>10: Reserved<br>11: Complementary mode, only the CH3 is valid for input, and the outputs of MCH3 and CH3 are complementary |
| 25:24 | MCH2MSEL[1:0] | Multi mode channel 2 mode select<br>00: Independent mode, MCH2 is independent of CH2<br>01: Reserved<br>10: Reserved<br>11: Complementary mode, only the CH2 is valid for input, and the outputs of MCH2 and CH2 are complementary |
| 23:22 | MCH1MSEL[1:0] | Multi mode channel 1 mode select<br>00: Independent mode, MCH1 is independent of CH1 |

| | | |
|---|---|---|
| | | 01: Reserved<br>10: Reserved<br>11: Complementary mode, only the CH1 is valid for input, and the outputs of MCH1 and CH1 are complementary |
| 21:20 | MCH0MSEL[1:0] | Multi mode channel 0 mode select<br>00: Independent mode, MCH0 is independent of CH0<br>01: Reserved<br>10: Reserved<br>11: Complementary mode, only the CH0 is valid for input, and the outputs of MCH0 and CH0 are complementary |
| 19 | DECDISDEN | Quadrature decoder signal disconnection detection enable<br>0: Quadrature decoder signal disconnection detection is disabled<br>1: Quadrature decoder signal disconnection detection is enabled |
| 18 | DECJDEN | Quadrature decoder signal jump (the two signals jump at the same time) detection enable<br>0: Quadrature decoder signal jump detection is disabled<br>1: Quadrature decoder signal jump detection is enabled |
| 17:16 | Reserved | Must be kept at reset value. |
| 15:14 | CH3OMPSEL[1:0] | Channel 3 output match pulse select<br>When the match events occur, this bit is used to select the output of O3CPRE which drives CH3_O.<br>00: The O3CPRE signal is output normally with the configuration of CH3COMCTL[2:0] bits.<br>01: Only the counter is counting up, the O3CPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.<br>10: Only the counter is counting down, the O3CPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.<br>11: Both the counter is counting up and counting down, the O3CPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle. |
| 13:12 | CH2OMPSEL[1:0] | Channel 2 output match pulse select<br>When the match events occur, this bit is used to select the output of O2CPRE which drives CH2_O.<br>00: The O2CPRE signal is output normal with the configuration of CH2COMCTL[2:0] bits.<br>01: Only when the counter is counting up, the O2CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Only when the counter is counting down, the O2CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>11: Both when the counter is counting up and counting down, the O2CPRE signal is output a pulse when the match events occurs, and the pulse width is one |

| | | |
|---|---|---|
| | | CK_TIMER clock cycle. |
| 11:10 | CH1OMPSEL[1:0] | Channel 1 output match pulse select<br>When the match events occur, this bit is used to select the output of O1CPRE which drives CH1_O.<br>00: The O1CPRE signal is output normal with the configuration of CH1COMCTL[2:0] bits.<br>01: Only when the counter is counting up, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Only when the counter is counting down, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>11: Both when the counter is counting up and counting down, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle. |
| 9:8 | CH0OMPSEL[1:0] | Channel 0 output match pulse select<br>When the match events occur, this bit is used to select the output of O0CPRE which drives CH0_O.<br>00: The O0CPRE signal is output normal with the configuration of CH0COMCTL[2:0] bits.<br>01: Only when the counter is counting up, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Only when the counter is counting down, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>11: Both when the counter is counting up and counting down, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle. |
| 7 | BRKENCH3 | Break control enable for channel 3<br>0: Disabled<br>1: Enabled |
| 6 | BRKENCH2 | Break control enable for channel 2<br>0: Disabled<br>1: Enabled |
| 5 | BRKENCH1 | Break control enable for channel 1<br>0: Disabled<br>1: Enabled |
| 4 | BRKENCH0 | Break control enable for channel 0<br>0: Disabled<br>1: Enabled |
| 3 | DTIENCH3 | Dead time inserted enable for channel 3<br>Enables the deadtime insertion in the outputs of MCH3_O and CH3_O. |

| | | |
|---|---|---|
| | | 0: Disabled<br>1: Enabled |
| 2 | DTIENCH2 | Dead time inserted enable for channel 2<br>Enables the deadtime insertion in the outputs of MCH2_O and CH2_O.<br>0: Disabled<br>1: Enabled |
| 1 | DTIENCH1 | Dead time inserted enable for channel 1<br>Enables the deadtime insertion in the outputs of MCH1_O and CH1_O.<br>0: Disabled<br>1: Enabled |
| 0 | DTIENCH0 | Dead time inserted enable for channel 0<br>Enables the deadtime insertion in the outputs of MCH0_O and CH0_O.<br>0: Disabled<br>1: Enabled |

### Free complementary channel protection register 0 (TIMERx_FCCHP0)

Address offset: 0x7C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

This register is used to configure the outputs of CH0_O/MCH0_O.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FCCHP0EN | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | ROS | IOS | Reserved | | DTCFG[7:0] | | | | | | | |
| | | | | rw | rw | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | FCCHP0EN | Free complementary channel protection register 0 enable<br>0: the ROS、IOS and DTCFG[7:0] bits in TIMERx_CCHP register is active<br>1: the ROS、IOS and DTCFG[7:0] bits in TIMERx_FCCHP0 register is active<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 30:12 | Reserved | Must be kept at reset value. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH0EN or CH0NEN bit is reset, the corresponding channel is output disabled. |

| | | |
|---|---|---|
| | | 1: “off-state” enabled. If the CH0EN or CH0NEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH0EN/CH0NEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CH0EN/CH0NEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:0 | DTCFG[7:0] | Dead time configure<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG [7:5] =3’b0xx: DTvalue = DTCFG [7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG [7:5] =3’b10x: DTvalue = (64+DTCFG [5:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*2.<br>DTCFG [7:5] =3’b110: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG [7:5] =3’b111: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*16.<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |

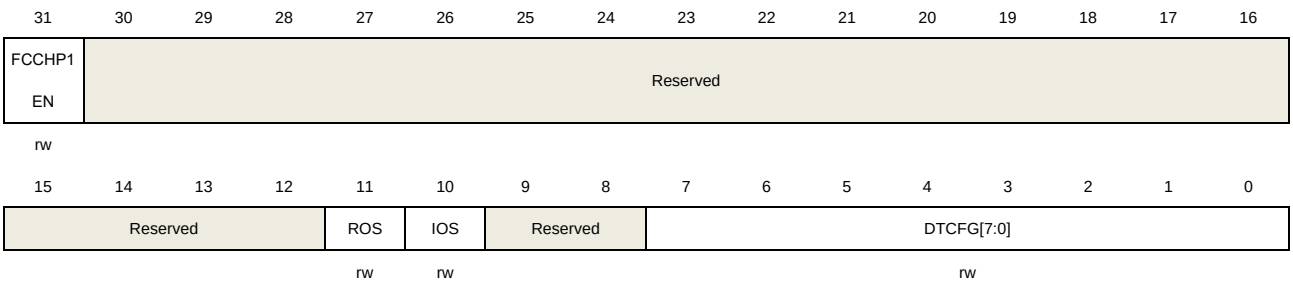
### Free complementary channel protection register 1 (TIMERx_FCCHP1)

Address offset: 0x80

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

This register is used to configure the outputs of CH1_O/MCH1_O.



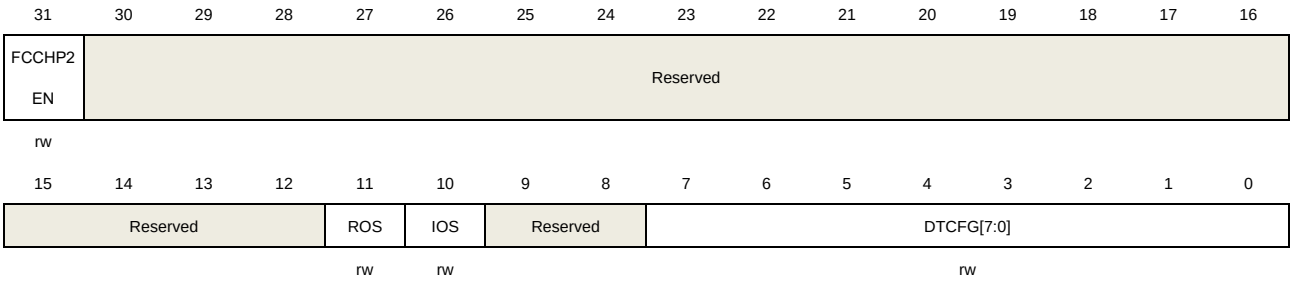

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | FCCHP1EN | Free complementary channel protection register 1 enable |

| | | |
|---|---|---|
| | | 0: the ROS、IOS and DTCFG[7:0] bits in TIMERx_CCHP register is active<br>1: the ROS、IOS and DTCFG[7:0] bits in TIMERx_FCCHP1 register is active<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 30:12 | Reserved | Must be kept at reset value. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH1EN or CH1NEN bit is reset, the corresponding channel is output disabled.<br>1: “off-state” enabled. If the CH1EN or CH1NEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH1EN/CH1NEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CH1EN/CH1NEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:0 | DTCFG[7:0] | Dead time configure<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG [7:5] =3’b0xx: DTvalue = DTCFG [7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG [7:5] =3’b10x: DTvalue = (64+DTCFG [5:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*2.<br>DTCFG [7:5] =3’b110: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG [7:5] =3’b111: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*16.<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |

### Free complementary channel protection register 2 (TIMERx_FCCHP2)

Address offset: 0x84
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).
This register is used to configure the outputs of CH2_O/MCH2_O.



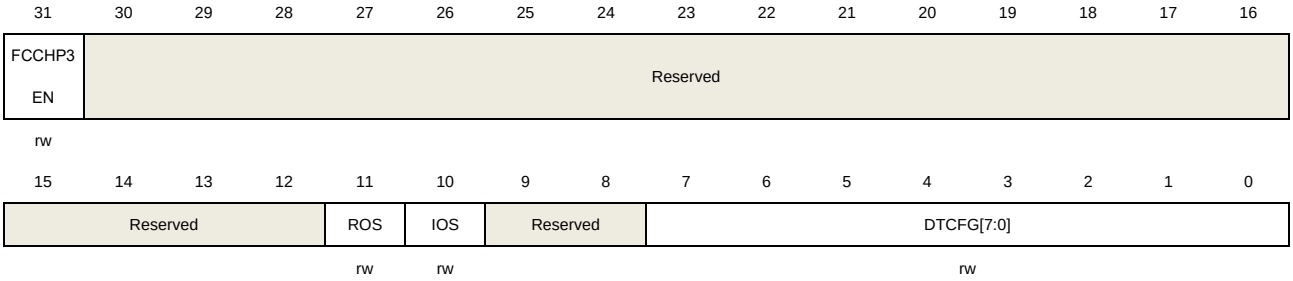

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | FCCHP2EN | Free complementary channel protection register 2 enable<br>0: the ROS、IOS and DTCFG[7:0] bits in TIMERx_CCHP register is active<br>1: the ROS、IOS and DTCFG[7:0] bits in TIMERx_FCCHP2 register is active<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 30:12 | Reserved | Must be kept at reset value. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH2EN or CH2NEN bit is reset, the corresponding channel is output disabled.<br>1: “off-state” enabled. If the CH2EN or CH2NEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH2EN/CH2NEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CH2EN/CH2NEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:0 | DTCFG[7:0] | Dead time configure<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG [7:5] =3’b0xx: DTvalue = DTCFG [7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG [7:5] =3’b10x: DTvalue = (64+DTCFG [5:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*2.<br>DTCFG [7:5] =3’b110: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG [7:5] =3’b111: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*16. |

This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

### Free complementary channel protection register 3 (TIMERx_FCCHP3)

Address offset: 0x88
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).
This register is used to configure the outputs of CH3_O/MCH3_O.




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | FCCHP3EN | Free complementary channel protection register 0 enable<br>0: the ROS、IOS and DTCFG[7:0] bits in TIMERx_CCHP register is active<br>1: the ROS、IOS and DTCFG[7:0] bits in TIMERx_FCCHP3 register is active<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |
| 30:12 | Reserved | Must be kept at reset value. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH3EN or CH3NEN bit is reset, the corresponding channel is output disabled.<br>1: “off-state” enabled. If the CH3EN or CH3NEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode.<br>0: “off-state” disabled. If the CH3EN/CH3NEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CH3EN/CH3NEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is |

| | | |
|---|---|---|
| | | 10 or 11. |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:0 | DTCFG[7:0] | Dead time configure<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:<br>DTCFG [7:5] =3'b0xx: DTvalue = DTCFG [7:0]x $t_{DT}$, $t_{DT}$=$t_{DTS}$.<br>DTCFG [7:5] =3'b10x: DTvalue = (64+DTCFG [5:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*2.<br>DTCFG [7:5] =3'b110: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$=$t_{DTS}$*8.<br>DTCFG [7:5] =3'b111: DTvalue = (32+DTCFG [4:0])x$t_{DT}$, $t_{DT}$ =$t_{DTS}$*16.<br>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00. |

### TIMER0 alternate function control register 0 (TIMER0_AFCTL0)

Address offset: 0x8C

Reset value: 0x0000 0007

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1P | BRK0CMP0P | Reserved | | | | | | BRK0IN2P | BRK0IN1P | BRK0IN0P |
| | | | | | rw | rw | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1EN | BRK0CMP0EN | BRK0HPDFEN | Reserved | | | | | BRK0IN2EN | BRK0IN1EN | BRK0IN0EN |
| | | | | | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | BRK0CMP1P | BREAK0 CMP1 input polarity<br>This bit is used to configure the CMP1 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: CMP1 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP1 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 25 | BRK0CMP0P | BREAK0 CMP0 input polarity<br>This bit is used to configure the CMP0 input polarity, and the specific polarity is |

| | | |
|---|---|---|
| | | determined by this bit and the BRK0P bit.<br>0: CMP0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 24:19 | Reserved | Must be kept at reset value. |
| 18 | BRK0IN2P | BREAK0 BRKIN2 alternate function input polarity<br>This bit is used to configure the BRKIN2 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN2 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN2 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 17 | BRK0IN1P | BREAK0 BRKIN1 alternate function input polarity<br>This bit is used to configure the BRKIN1 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN1 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN1 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 16 | BRK0IN0P | BREAK0 BRKIN0 alternate function input polarity<br>This bit is used to configure the BRKIN0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | BRK0CMP1EN | BREAK0 CMP1 enable<br>0: CMP1 input disabled<br>1: CMP1 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

| | | |
|---|---|---|
| 9 | BRK0CMP0EN | BREAK0 CMP0 enable<br>0: CMP0 input disabled<br>1: CMP0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 8 | BRK0HPDFEN | BREAK0 HPDF input(hpdf_break[0]) enable<br>0: HPDF input disabled<br>1: HPDF input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | BRK0IN2EN | BREAK0 BRKIN2 alternate function input enable<br>0: BRKIN2 alternate function input disabled<br>1: BRKIN2 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 1 | BRK0IN1EN | BREAK0 BRKIN1 alternate function input enable<br>0: BRKIN1 alternate function input disabled<br>1: BRKIN1 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 0 | BRK0IN0EN | BREAK0 BRKIN0 alternate function input enable<br>0: BRKIN0 alternate function input disabled<br>1: BRKIN0 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### TIMER0 alternate function control register 1 (TIMER0_AFCTL1)

Address offset: 0x90

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK1CMP1P | BRK1CMP0P | Reserved | | | | | | BRK1IN2P | BRK1IN1P | BRK1IN0P |
| | | | | | rw | rw | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK1CMP1EN | BRK1CMP0EN | BRK1HPDFEN | Reserved | | | | | BRK1IN2EN | BRK1IN1EN | BRK1IN0EN |
| | | | | | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | BRK1CMP1P | BREAK1 CMP1 input polarity<br>This bit is used to configure the CMP1 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: CMP1 input signal will not be inverted (BRK1P =0, the input signal is active low; BRK1P =1, the input signal is active high)<br>1: CMP1 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 25 | BRK1CMP0P | BREAK1 CMP0 input polarity<br>This bit is used to configure the CMP0 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: CMP0 input signal will not be inverted (BRK1P =0, the input signal is active low; BRK1P =1, the input signal is active high)<br>1: CMP0 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 24:19 | Reserved | Must be kept at reset value. |
| 18 | BRK1IN2P | BREAK1 BRKIN2 alternate function input polarity<br>This bit is used to configure the BRKIN2 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: BRKIN2 input signal will not be inverted (BRK1P =0, the input signal is active low; BRKP =1, the input signal is active high)<br>1: BRKIN2 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 17 | BRK1IN1P | BREAK1 BRKIN1 alternate function input polarity<br>This bit is used to configure the BRKIN1 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: BRKIN1 input signal will not be inverted (BRK1P =0, the input signal is active low; BRKP =1, the input signal is active high)<br>1: BRKIN1 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 16 | BRK1IN0P | BREAK1 BRKIN0 alternate function input polarity<br>This bit is used to configure the BRKIN0 input polarity, and the specific polarity is |

| | | |
|---|---|---|
| | | determined by this bit and the BRK1P bit.<br>0: BRKIN0 input signal will not be inverted (BRK1P =0, the input signal is active low; BRKP =1, the input signal is active high)<br>1: BRKIN0 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | BRK1CMP1EN | BREAK1 CMP1 enable<br>0: CMP1 input disabled<br>1: CMP1 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 9 | BRK1CMP0EN | BREAK1 CMP0 enable<br>0: CMP0 input disabled<br>1: CMP0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 8 | BRK1HPDFEN | BREAK1 HPDF input(hpdf_break[1]) enable<br>0: HPDF input disabled<br>1: HPDF input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | BRK1IN2EN | BREAK1 BRKIN2 alternate function input enable<br>0: BRKIN2 alternate function input disabled<br>1: BRKIN2 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 1 | BRK1IN1EN | BREAK1 BRKIN1 alternate function input enable<br>0: BRKIN1 alternate function input disabled<br>1: BRKIN1 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 0 | BRK1IN0EN | BREAK1 BRKIN0 alternate function input enable<br>0: BRKIN0 alternate function input disabled<br>1: BRKIN0 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### TIMER7 alternate function control register 0 (TIMER7_AFCTL0)

Address offset: 0x8C

Reset value: 0x0000 0007

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1P | BRK0CMP0P | Reserved | | | | | | BRK0IN2P | BRK0IN1P | BRK0IN0P |
| | | | | | rw | rw | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1EN | BRK0CMP0EN | BRK0HPDFEN | Reserved | | | | | BRK0IN2EN | BRK0IN1EN | BRK0IN0EN |
| | | | | | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | BRK0CMP1P | BREAK0 CMP1 input polarity<br>This bit is used to configure the CMP1 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: CMP1 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP1 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 25 | BRK0CMP0P | BREAK0 CMP0 input polarity<br>This bit is used to configure the CMP0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: CMP0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 24:19 | Reserved | Must be kept at reset value. |
| 18 | BRK0IN2P | BREAK0 BRKIN2 alternate function input polarity<br>This bit is used to configure the BRKIN2 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN2 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN2 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low) |

| | | |
|---|---|---|
| | | This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 17 | BRK0IN1P | BREAK0 BRKIN1 alternate function input polarity<br>This bit is used to configure the BRKIN1 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN1 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN1 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 16 | BRK0IN0P | BREAK0 BRKIN0 alternate function input polarity<br>This bit is used to configure the BRKIN0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | BRK0CMP1EN | BREAK0 CMP1 enable<br>0: CMP1 input disabled<br>1: CMP1 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 9 | BRK0CMP0EN | BREAK0 CMP0 enable<br>0: CMP0 input disabled<br>1: CMP0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 8 | BRK0HPDFEN | BREAK0 HPDF input(hpdf_break[0]) enable<br>0: HPDF input disabled<br>1: HPDF input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | BRK0IN2EN | BREAK0 BRKIN2 alternate function input enable<br>0: BRKIN2 alternate function input disabled<br>1: BRKIN2 alternate function input enabled |

| | | |
|---|---|---|
| | | This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 1 | BRK0IN1EN | BREAK0 BRKIN1 alternate function input enable<br>0: BRKIN1 alternate function input disabled<br>1: BRKIN1 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 0 | BRK0IN0EN | BREAK0 BRKIN0 alternate function input enable<br>0: BRKIN0 alternate function input disabled<br>1: BRKIN0 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### TIMER7 alternate function control register 1 (TIMER7_AFCTL1)

Address offset: 0x90

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK1CMP1P | BRK1CMP0P | Reserved | | | | | | BRK1IN2P | BRK1IN1P | BRK1IN0P |
| | | | | | rw | rw | | | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK1CMP1EN | BRK1CMP0EN | BRK1HPDFEN | Reserved | | | | | BRK1IN2EN | BRK1IN1EN | BRK1IN0EN |
| | | | | | rw | rw | rw | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | BRK1CMP1P | BREAK1 CMP1 input polarity<br>This bit is used to configure the CMP1 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: CMP1 input signal will not be inverted (BRK1P =0, the input signal is active low; BRK1P =1, the input signal is active high)<br>1: CMP1 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 25 | BRK1CMP0P | BREAK1 CMP0 input polarity<br>This bit is used to configure the CMP0 input polarity, and the specific polarity is determined by this bit and the BRK1P bit. |

| | | |
|---|---|---|
| | | 0: CMP0 input signal will not be inverted (BRK1P =0, the input signal is active low; BRK1P =1, the input signal is active high)<br>1: CMP0 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 24:19 | Reserved | Must be kept at reset value. |
| 18 | BRK1IN2P | BREAK1 BRKIN2 alternate function input polarity<br>This bit is used to configure the BRKIN2 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: BRKIN2 input signal will not be inverted (BRK1P =0, the input signal is active low; BRKP =1, the input signal is active high)<br>1: BRKIN2 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 17 | BRK1IN1P | BREAK1 BRKIN1 alternate function input polarity<br>This bit is used to configure the BRKIN1 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: BRKIN1 input signal will not be inverted (BRK1P =0, the input signal is active low; BRKP =1, the input signal is active high)<br>1: BRKIN1 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 16 | BRK1IN0P | BREAK1 BRKIN0 alternate function input polarity<br>This bit is used to configure the BRKIN0 input polarity, and the specific polarity is determined by this bit and the BRK1P bit.<br>0: BRKIN0 input signal will not be inverted (BRK1P =0, the input signal is active low; BRKP =1, the input signal is active high)<br>1: BRKIN0 input signal will be inverted (BRK1P =0, the input signal is active high; BRK1P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | BRK1CMP1EN | BREAK1 CMP1 enable<br>0: CMP1 input disabled<br>1: CMP1 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

| | | |
|---|---|---|
| 9 | BRK1CMP0EN | BREAK1 CMP0 enable<br>0: CMP0 input disabled<br>1: CMP0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 8 | BRK1HPDFEN | BREAK1 HPDF input(hpdf_break[1]) enable<br>0: HPDF input disabled<br>1: HPDF input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | BRK1IN2EN | BREAK1 BRKIN2 alternate function input enable<br>0: BRKIN2 alternate function input disabled<br>1: BRKIN2 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 1 | BRK1IN1EN | BREAK1 BRKIN1 alternate function input enable<br>0: BRKIN1 alternate function input disabled<br>1: BRKIN1 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 0 | BRK1IN0EN | BREAK1 BRKIN0 alternate function input enable<br>0: BRKIN0 alternate function input disabled<br>1: BRKIN0 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### Watchdog counter period register(TIMERx_WDGPER)

Address offset: 0x94

Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDGPER[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDGPER[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | WDGPER[31:0] | Watchdog counter period value |

This register contains the period of the two watchdog counter. When the counters countius to count to this value, the counter will timeout and the interrupt flag DECDISIF is set. If DECDISIE=1, the corresponding interrupt is generated.

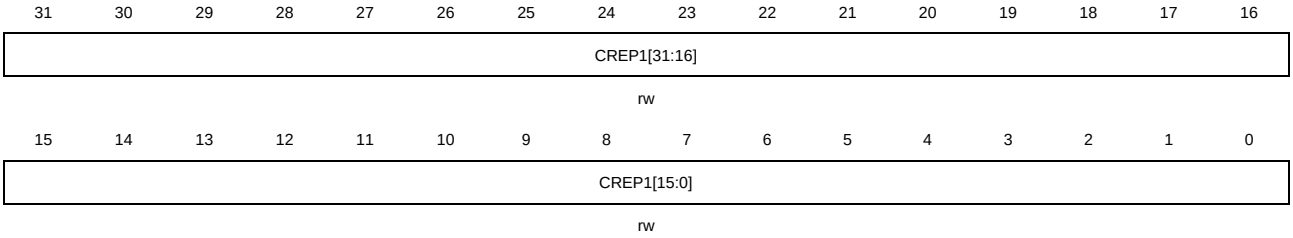
Note: This register is just used in quadrature decoder signal disconnection detection function(with DECDISDEN =1).

### Counter repetition register 1 (TIMERx_CREP1)

Address offset: 0x98

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



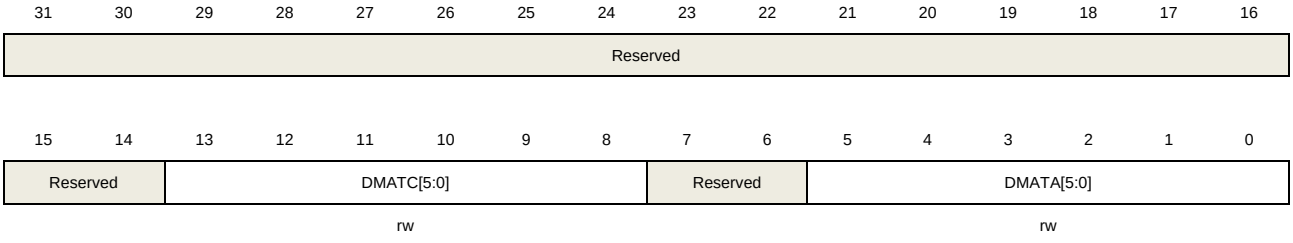

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CREP1[31:0] | Counter repetition value 1<br>This bit-field is 32 bits and can be read on the fly.<br>This bit-field specifies the update event generation rate. Each time the repetition counter counts down to zero, an update event will be generated. The update rate of the shadow registers is also affected by this bit-field when these shadow registers are enabled.<br>**Note**: This bit-field just used with CREPSEL =1(in TIMERx_CFG register). |

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0xE0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:8 | DMATC[5:0] | DMA transfer count |

| | | |
|---|---|---|
| | | This field defines the times of accessing(R/W) the TIMERx_DMATB register by DMA.<br>6'b000000: transfer 1 time<br>6'b000001: transfer 2 times<br>…<br>6'b100101: transfer 38 times |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:0 | DMATA[5:0] | DMA transfer access start address<br>This field defines the start address of accessing the TIMERx_DMATB register by DMA. When the first access to the TIMERx_DMATB register is done, this bit-field specifies the address just accessed. And then the address of the second access to the TIMERx_DMATB register will be (start address + 0x4).<br>6'b000000: TIMERx_CTL0<br>6'b000001: TIMERx_CTL1<br>…<br>6'b100101: TIMERx_CREP1<br>In a word: start address = TIMERx_CTL0 + DMATA*4 |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0xE4
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

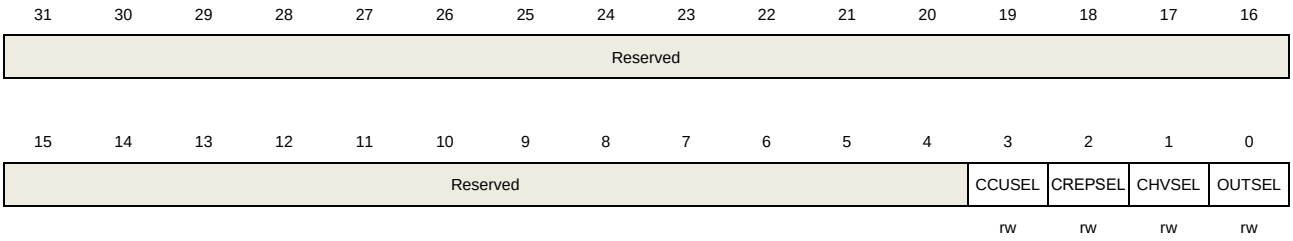
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMATB[31:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address ranges from (start address) to (start address + transfer count * 4) will be accessed.<br>The transfer count is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | CCUSEL | Commutation control shadow register update select<br>This bit is valid only when the CCUC[2:0] bit-field are set to 100, 101 and 110.<br>0: The shadow registers update when the counter generates an overflow/ underflow event.<br>1: The shadow registers update when the counter generates an overflow/ underflow event and the repetition counter value is zero. |
| 2 | CREPSEL | The counter repetition register select<br>This bit is used to select the counter repetition register.<br>0: The update event rate is depended to TIMERx_CREP0 register<br>1: The update event rate is depended to TIMERx_CREP1 register |
| 1 | CHVSEL | Write CHxVAL register selection bit<br>This bit-field is set and reset by software.<br>1: If the value to be written to the CHxVAL register is the same as the value of CHxVAL register, the write access is ignored.<br>0: No effect. |
| 0 | OUTSEL | The output value selection bit<br>This bit-field is set and reset by software.<br>1: If POEN bit and IOS bit are 0, the output is disabled.<br>0: No effect. |

## 24.2. General level0 timer (TIMERx, x=1,2,3,4,22,23,30,31)

### 24.2.1. Overview

The general level0 timer module (TIMER1/2/3/4/22/23/30/31) is a four-channel timer that supports input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level0 timer has a 16-bit or 32-bit counter that can be used as an unsigned counter.

In addition, the general level0 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counter value increasing in unison.

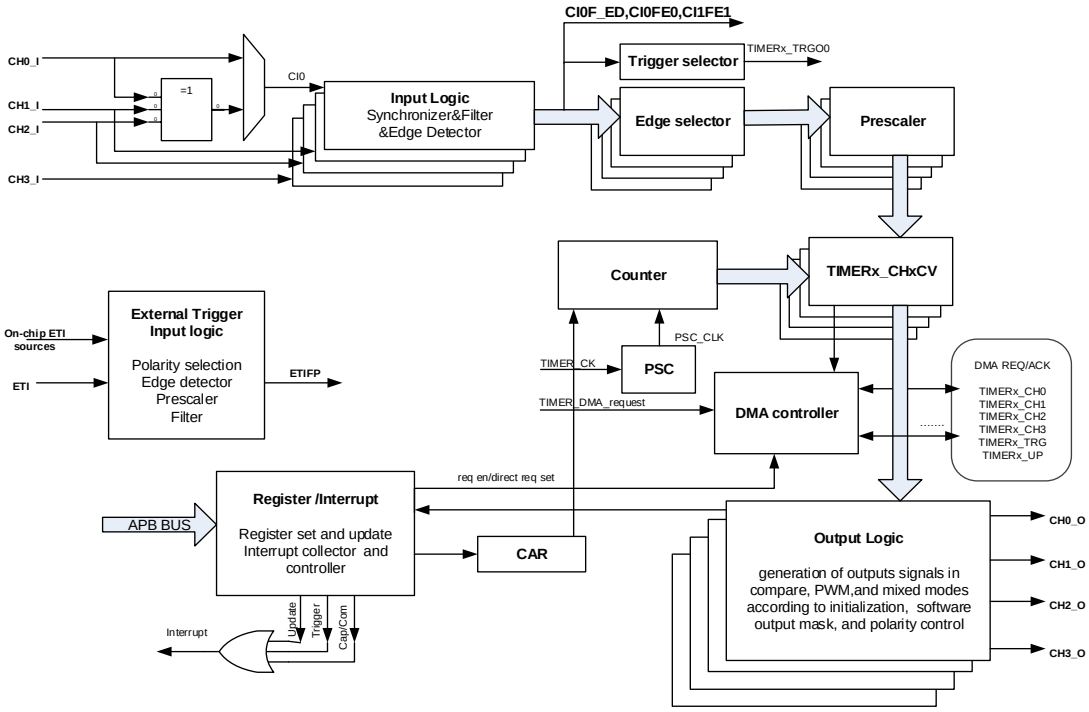
### 24.2.2. Characteristics

- Total channel num: 4.
- Counter width: 16 bits(TIMER2/3/30/31) or 32 bits(TIMER1/4/22/23).
- Selectable clock source: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: up counting, down counting and center-aligned counting.
- Quadrature decoder: used for motion tracking and determination of both rotation direction and position.
- Hall sensor function: used for 3-phase motor control.
- Programmable prescaler: 16 bits. The factor can be changed ongoing.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode and single pulse mode.
- Auto reload function.
- Interrupt output or DMA request: update event, trigger event and compare/capture event.
- Daisy chaining of timer module allows a single timer to start multiple timers.
- Timer synchronization allows the selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 24.2.3. Block diagram

*Figure 24-50. General Level 0 timer block diagram* provides details on the internal configuration of the general level 0 timer.

**Figure 24-50. General Level 0 timer block diagram**




## 24.2.4. Function overview

### Clock selection

The general level0 TIMER has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by TSCFGy[4:0] (y=0..9,15) in SYSCFG_TIMERxCFG(x=1..4,22,23,30,31) registers.

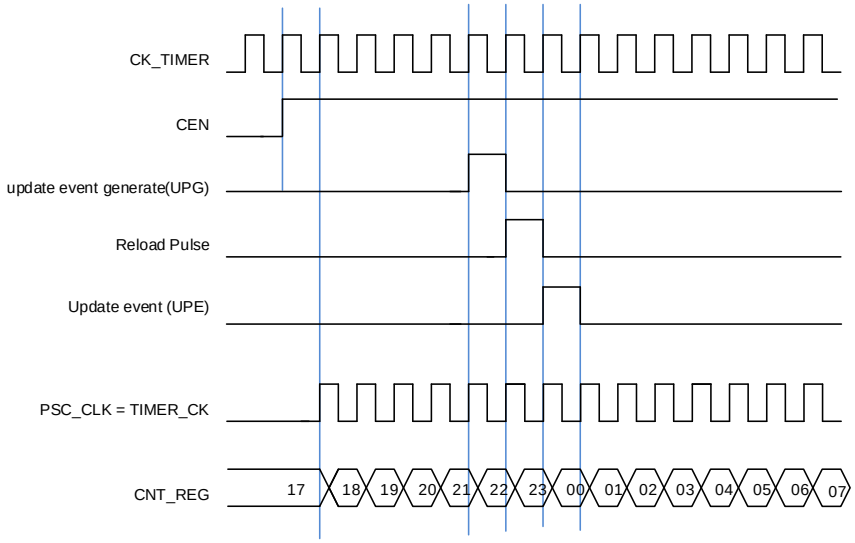
- TSCFGy[4:0] (y=0..9,15) in SYSCFG_TIMERxCFG(x=1..4,22,23,30,31) registers. Internal clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when TSCFGy[4:0] (y=0..9,15) = 5'b00000 in SYSCFG_TIMERxCFG(x=1..4,22,23,30,31) registers. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK which drives counter's prescaler to count is equal to CK_TIMER which is from RCU module.

If TSCFGy[4:0] (y=0..2,6,8,9) in SYSCFG_TIMERxCFG(x=1..4,22,23,30,31) registers are setting to an available value, the prescaler is clocked by other clock sources selected in the TSCFGy[4:0] (y=0..2,6,8,9) bit-filed, more details will be introduced later. When the TSCFGy[4:0] (y=3,4,5,7) are setting to an available value, the internal clock TIMER_CK is the counter prescaler driving clock source.

**Figure 24-51. Normal mode, internal clock divided by 1**




- TSCFG6[4:0] are setting to a nonzero value (external clock mode 0). External input pin is selected as timer clock source.

The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CH0/TIMERx_CH1. This mode can be selected by setting TSCFG6[4:0] to 0x5~0x7.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0~ITI14. This mode can be selected by setting TSCFG6[4:0] to 0x1~0x4, 0x9~0x14.

- SMC1= 1'b1 (external clock mode 1). External input ETI is selected as timer clock source.

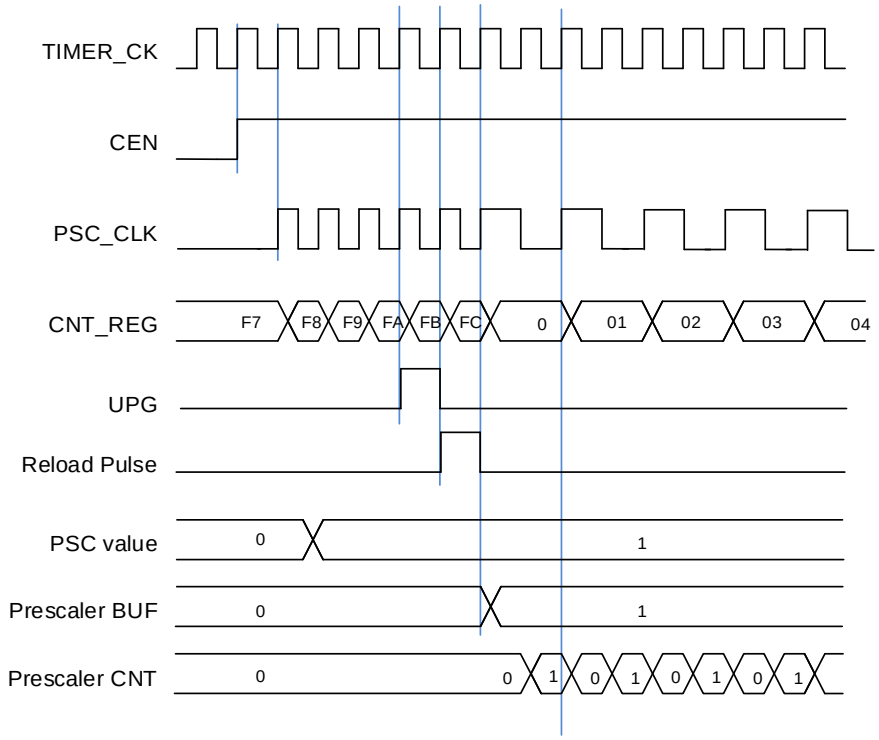
The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx_SMCFG register to 1. The other way to select the ETI signal as the clock source is setting the TSCFG6[4:0] to 0x8. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as the clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

**Note:** The ETI signal can be input from an external ETI pin or provide by on-chip peripherals, please refer to ***Trigger selection for TIMER1_ETI register (TRIGSEL_TIMER1ETI)*** for more details.

### Clock prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor ranging from 1 to 65536. It is controlled by prescaler register (TIMERx_PSC) which can be changed ongoing, but it is adopted at the next update event.

**Figure 24-52. Counter timing diagram with prescaler division change from 1 to 2**




### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. The update event is generated each time when counter overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (auto reload register, prescaler register) are updated.

***Figure 24-53. Timing chart of up counting mode, PSC=0/2*** and ***Figure 24-54. Timing chart of up counting, change TIMERx_CAR ongoing*** show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 24-53. Timing chart of up counting mode, PSC=0/2**

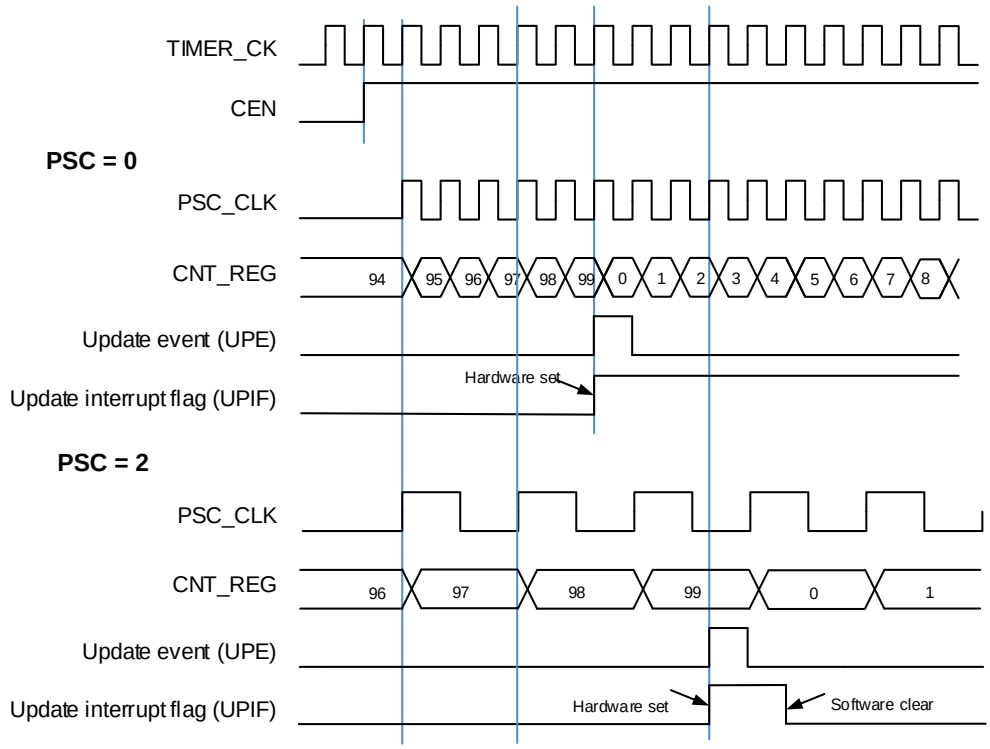
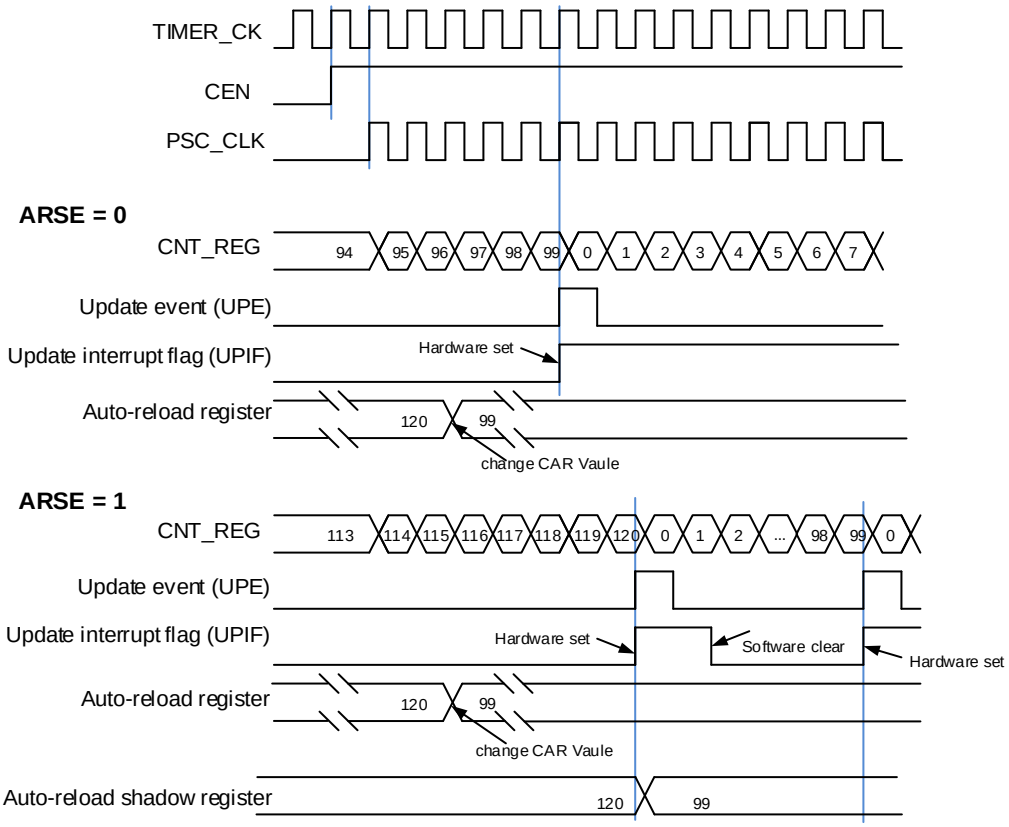



**Figure 24-54. Timing chart of up counting, change TIMERx_CAR ongoing**



### Down counting mode

In this mode, the counter counts down continuously from the counter reload value, which is defined in the TIMERx_CAR register, in a count-down direction. Once the counter reaches 0, the counter restarts to count again from the counter reload value. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 1 for the down counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to the counter reload value and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (auto reload register, prescaler register) are updated.

*Figure 24-55. Timing chart of down counting mode, PSC=0/2* and *Figure 24-56. Timing chart of down counting mode, change TIMERx_CAR ongoing* show some examples of the counter behavior for different clock frequencies when TIMERx_CAR = 0x99.

**Figure 24-55. Timing chart of down counting mode, PSC=0/2**

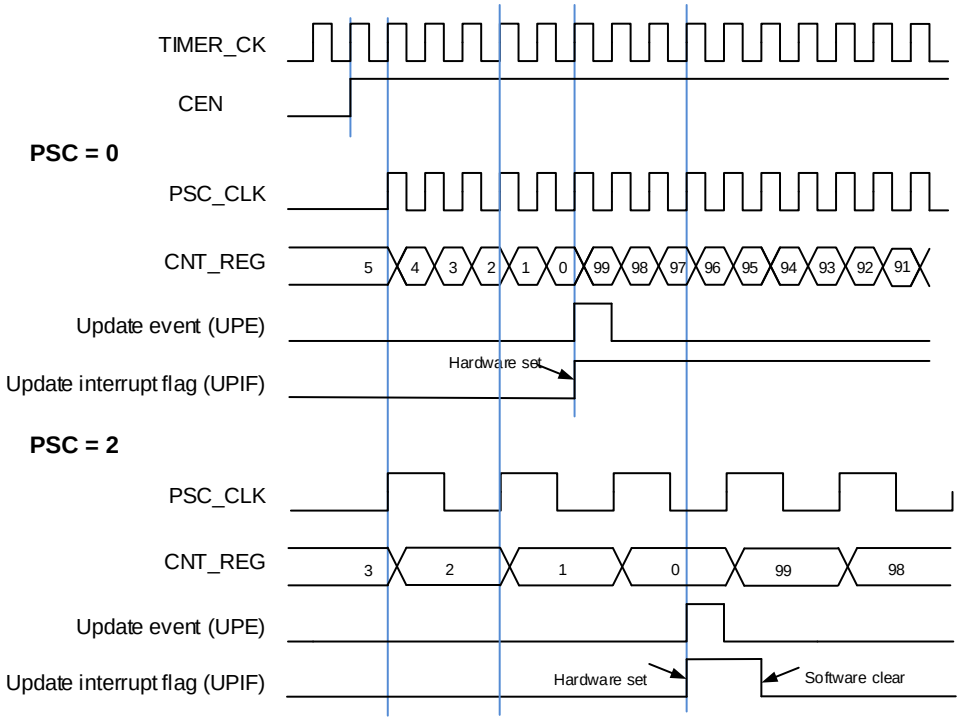
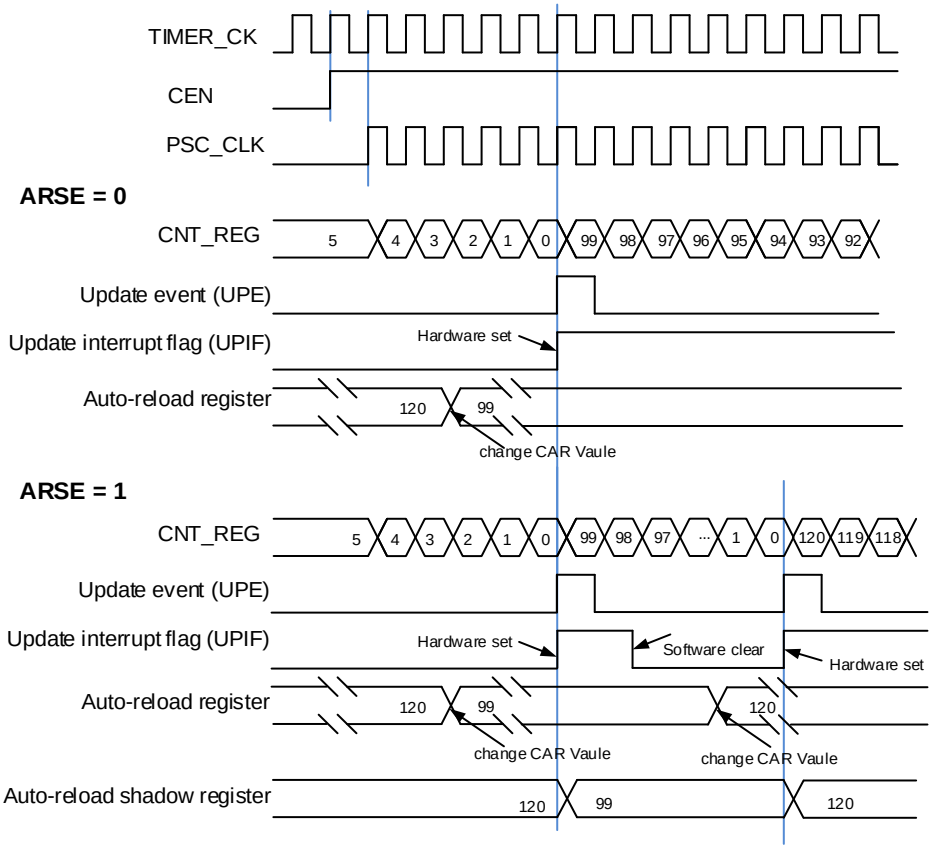


**Figure 24-56. Timing chart of down counting mode, change TIMERx_CAR ongoing**




### Center-aligned counting mode

In the center-aligned counting mode, the counter counts up from 0 to the counter reload value and then counts down to 0 alternatively. The timer module generates an overflow event when the counter counts to (TIMERx_CAR-1) in the count-up direction and generates an underflow event when the counter counts to 1 in the count-down direction. The counting direction bit DIR in the TIMERx_CTL0 register is read-only and indicates the counting direction when in the center-aligned counting mode. The counting direction is updated by hardware automatically.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 and generate an update event irrespective of whether the counter is counting up or down in the center-aligned counting mode.

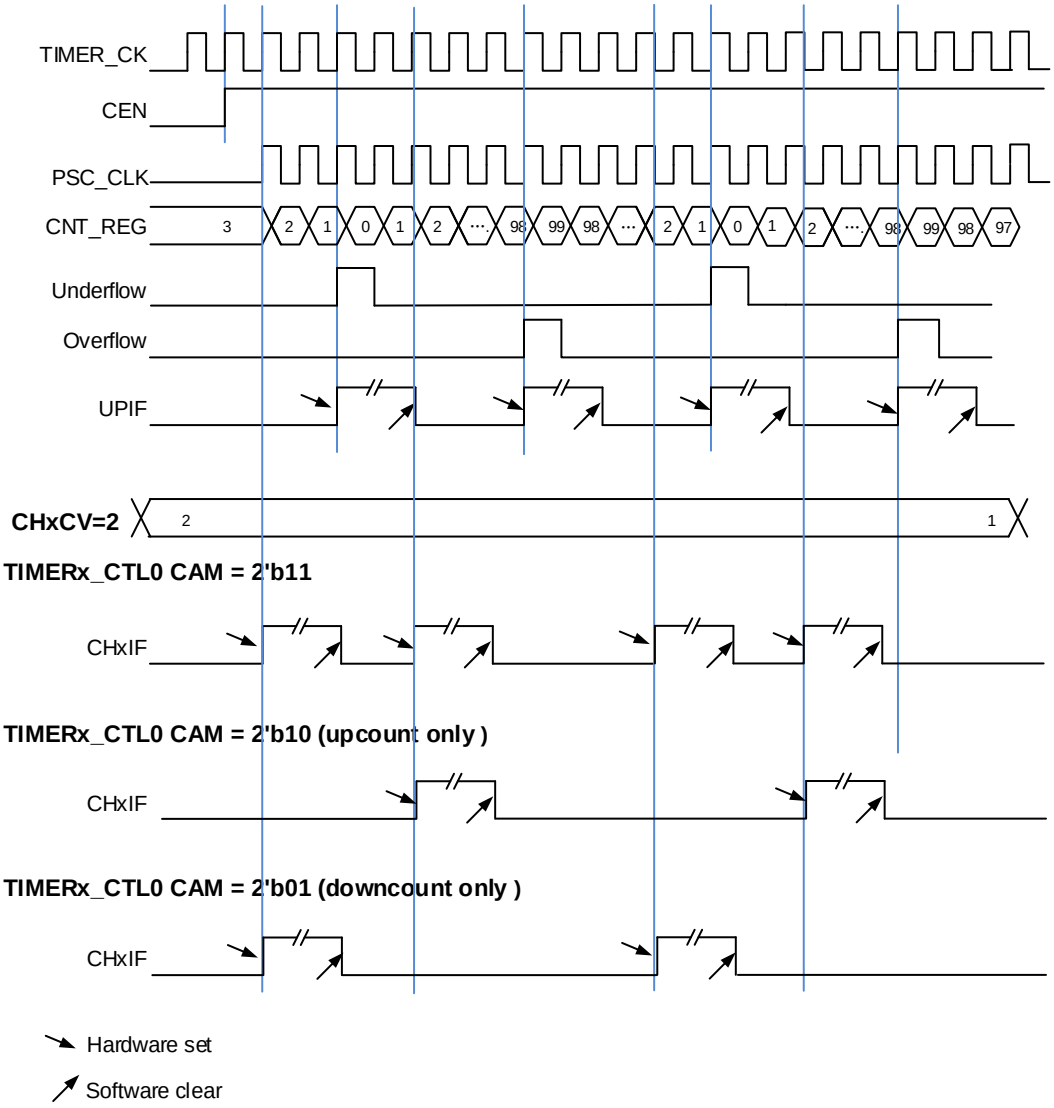
The UPIF bit in the TIMERx_INTF register will be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 24-57. Timing chart of center-aligned counting mode***.

If the UPDIS bit in the TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (auto-reload register, prescaler register) are updated. ***Figure 24-57. Timing chart of center-aligned counting mode*** shows the example

of the counter behavior when TIMERx_CAR=0x99, TIMERx_PSC=0x0.

**Figure 24-57. Timing chart of center-aligned counting mode**



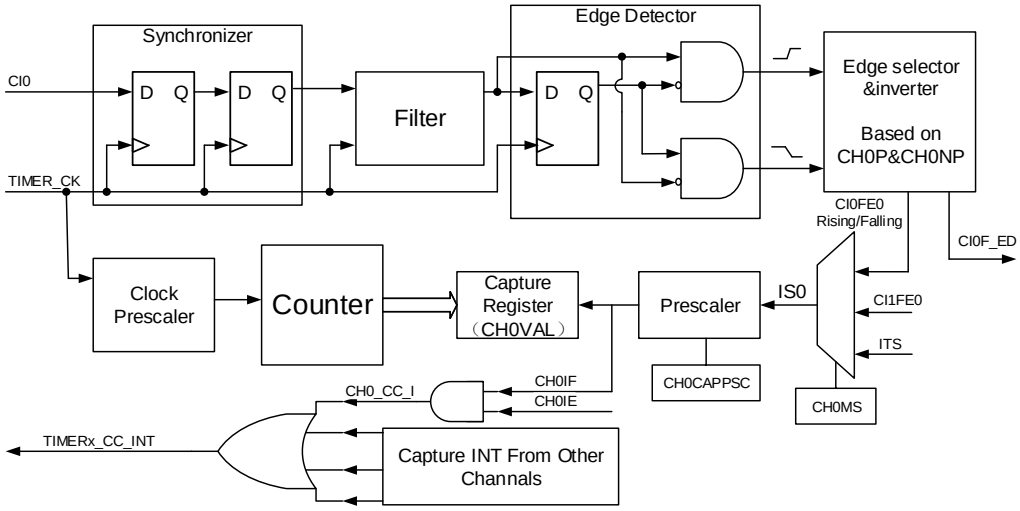

## Capture/compare channels

The general level0 timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- **Input capture mode**

Input capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if it is enabled when CHxIE=1.

**Figure 24-58. Input capture logic**




The input signals of channelx (CIx) can be the TIMERx_CHx signal or the XOR signal of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals (just for CI0). First, the input signal of channel (CIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP bit. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT in TIMERx_CHCTL0).
Based on the input signal and quality of requested signal, configure compatible CHxCAPFLT.

**Step2**: Edge selection (CHxP and CHxNP bits in TIMERx_CHCTL2).
Rising edge or falling edge, choose one by configuring CHxP and CHxNP bits.

**Step3**: Capture source selection (CHxMS in TIMERx_CHCTL0)
As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x0) and TIMERx_CHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN in TIMERx_DMAINTEN)
Enable the related interrupt to get the interrupt and DMA request.

**Step5:** Capture enable (CHxEN in TIMERx_CHCTL2)

**Result**: When the wanted input signal is captured, TIMERx_CHxCV will be set by counter's value and CHxIF is asserted. If the CHxIF is 1, the CHxOF will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN.

**Direct generation**: A DMA request or interrupt is generated by setting CHxG directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0

capture signals by setting CH0MS to 3'b001 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 3'b010 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty cycle.

- **Output compare mode**

**Figure 24-59. Output compare logic (x=0,1,2,3)**

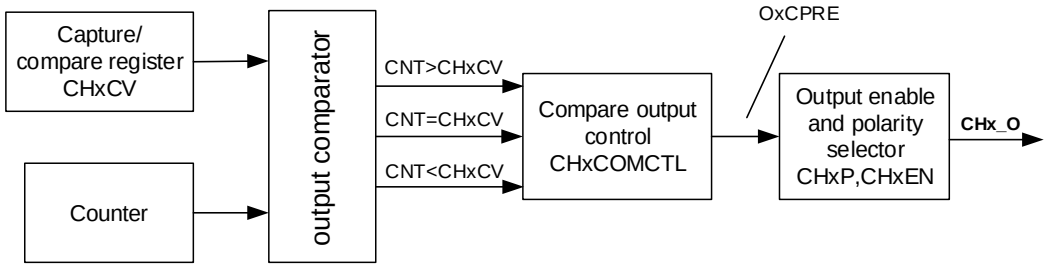



*Figure 24-59. Output compare logic (x=0,1,2,3)* shows the logic circuit of output compare mode. The relationship between the channel output signal CHx_O and the OxCPRE signal (more details refer to *Channel output prepare signal*) is described as blew: The active level of O0CPRE is high, the output level of CH0_O depends on OxCPRE signal, CHxP bit and CH0P bit (please refer to the TIMERx_CHCTL2 register for more details).For example, configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled),

If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;

If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the TIMERx_CHxCV register, the CHxIF bit will be set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, if CHxDEN=1.

So, the process can be divided into several steps as below:

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL.
- Select the active polarity by CHxP.
- Enable the output by CHxEN.

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CHxDEN.

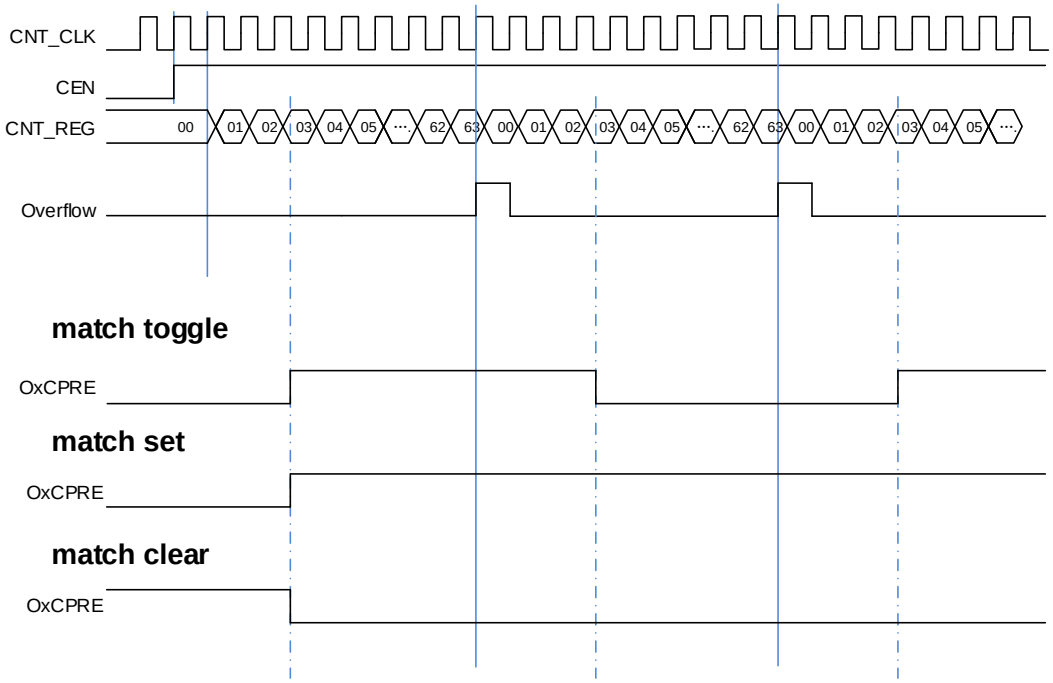
**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.

The TIMERx_CHxCV can be changed onging to meet the expected waveform.

**Step5:** Start the counter by configuring CEN to 1.

*Figure 24-60. Output-compare under three modes* shows the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 24-60. Output-compare under three modes**




### PWM mode

In the PWM output mode (by setting the CHxCOMCTL bit to 4'b0110 (PWM mode 0) or to 4'b0111 (PWM mode 1)), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

Based on the counter mode, PWM can also be divided into EAPWM (Edge-aligned PWM) and CAPWM (Center-aligned PWM).

The EAPWM's period is determined by TIMERx_CAR and the duty cycle is determined by TIMERx_CHxCV. *Figure 24-61. Timing chart of EAPWM* shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2*TIMERx_CAR, and duty cycle is determined by 2*TIMERx_CHxCV. *Figure 24-62. Timing chart of CAPWM* shows the CAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx_CHxCV is greater than the value of TIMERx_CAR, the output will be always active in PWM mode 0 (CHxCOMCTL=4'b0110). And if the value of TIMERx_CHxCV is greater than the value of TIMERx_CAR, the output will be always inactive in PWM mode 1 (CHxCOMCTL=4'b0111).

**Figure 24-61. Timing chart of EAPWM**

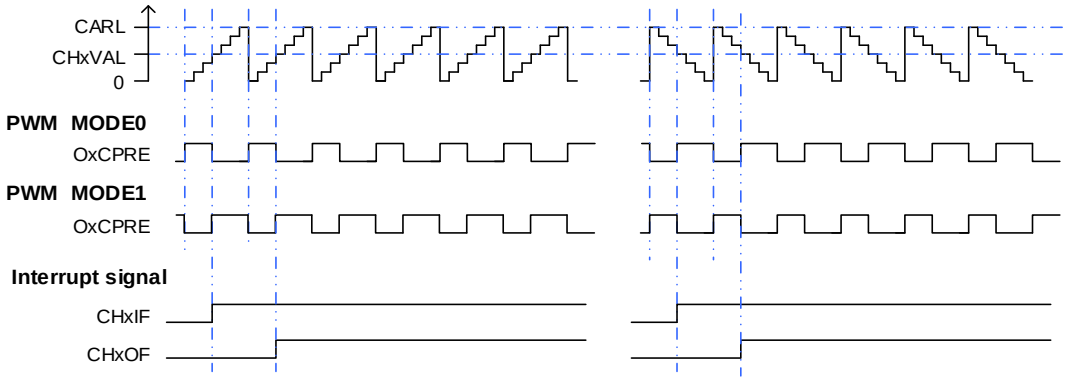
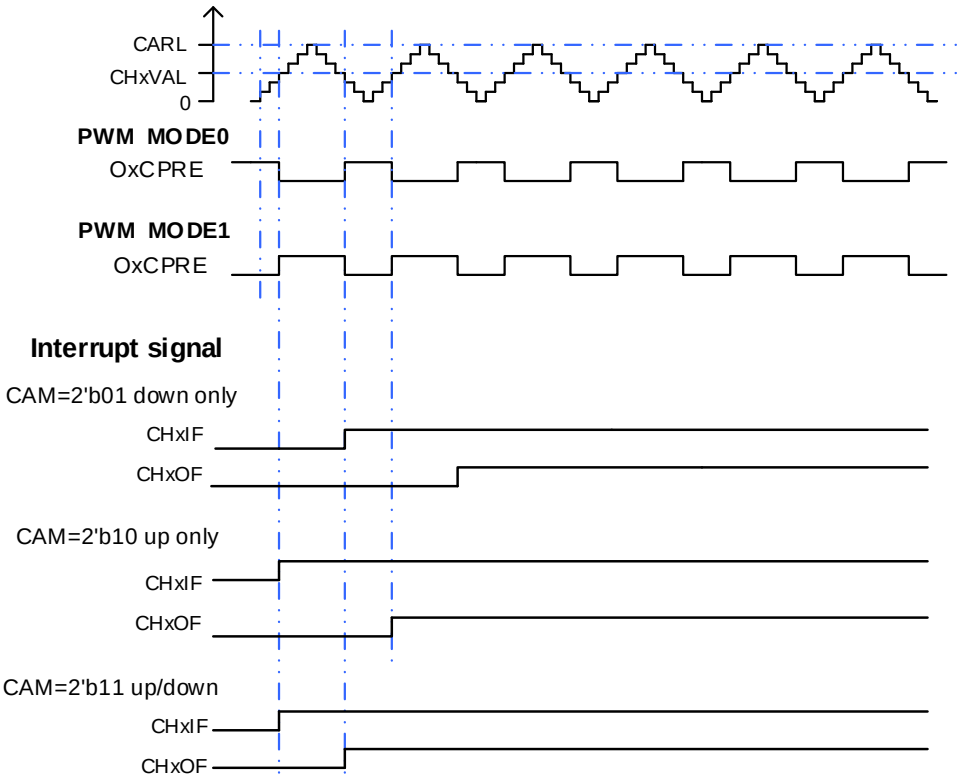



**Figure 24-62. Timing chart of CAPWM**




## Composite PWM mode

In the Composite PWM mode (CHxCPWMEN = 1'b1, CHxMS[2:0] = 3'b000 and CHxCOMCTL = 4'b0110 or 4'b0111), the PWM signal output in channel x (x=0..3) is composited by CHxVAL and CHxCOMVAL_ADD bits.

If CHxCOMCTL = 4'b0110 (PWM mode 0) and DIR = 1'b0 (up counting mode), or CHxCOMCTL = 4'b0111 (PWM mode 1) and DIR = 1'b1 (Down counting mode), the channel x output is forced low when the counter matches the value of CHxVAL. It is forced high when the counter matches the value of CHxCOMVAL_ADD.

If CHxCOMCTL = 4'b0111 (PWM mode 1) and DIR = 1'b0 (up counting mode), or

CHxCOMCTL = 4'b0110 (PWM mode 0) and DIR = 1'b1 (down counting mode) the channel x output is forced high when the counter matches the value of CHxVAL. It is forced low when the counter matches the value of CHxCOMVAL_ADD.

The PWM period is determined by (CARL + 0x0001) and the PWM pulse width is determined by the following table.

**Table 24-10.The Composite PWM pulse width**

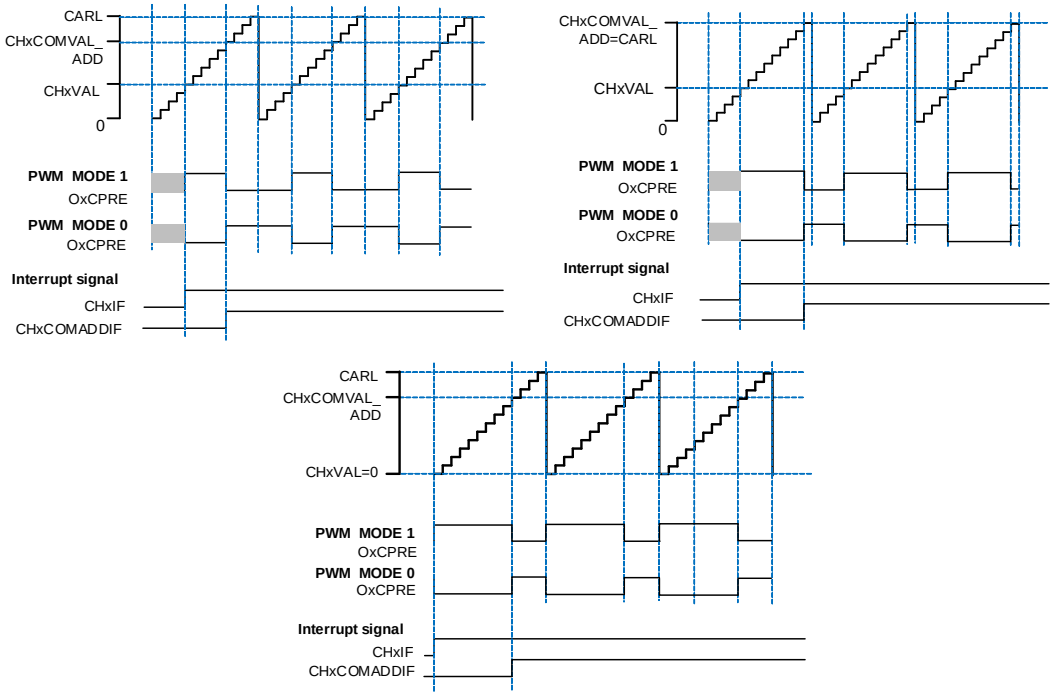
| Condition | Mode | PWM pulse width |
|---|---|---|
| CHxVAL < CHxCOMVAL_ADD ≤ CARL | PWM mode 0 | (CARL + 0x0001) + (CHxVAL – CHxCOMVAL_ADD) |
| | PWM mode 1 | (CHxCOMVAL_ADD – CHxVAL) |
| CHxCOMVAL_ADD < CHxVAL ≤ CARL | PWM mode 0 | (CHxVAL - CHxCOMVAL_ADD) |
| | PWM mode 1 | (CARL + 0x0001) + (CHxCOMVAL_ADD – CHxVAL) |
| (CHxVAL = CHxCOMVAL_ADD ≤ CARL) or (CHxVAL > CARL > CHxCOMVAL_ADD) | PWM mode 0 (up counting) or PWM mode 1 (down counting) | 100% |
| | PWM mode 0 (down counting) or PWM mode 1 (up counting) | 0% |
| CHxCOMVAL_ADD > CARL > CHxVAL | PWM mode 0(up counting) or PWM mode 1(down counting) | 0% |
| | PWM mode 0(down counting) or PWM mode 1(up counting) | 100% |
| (CHxVAL>CARL) and (CHxCOMVAL_ADD > CARL) | - | The output of CHx_O is keeping |

When the counter reaches the value of CHxVAL, the CHxIF bit is set and the channel x interrupt is generated if CHxIE = 1, and the DMA request will be asserted, if CHxDEN=1. When the counter reaches the value of CHxCOMVAL_ADD, the CHxCOMADDIF bit is set (this flag just used in composite PWM mode, when CHxCPWMEN=1) and the channel x additional compare interrupt is generated if CHxCOMADDIE = 1 (Only interrupt is generated, no DMA request is generated).

According to the relationship among CHxVAL, CHxCOMVAL_ADD and CARL, it can be divided into four situations:

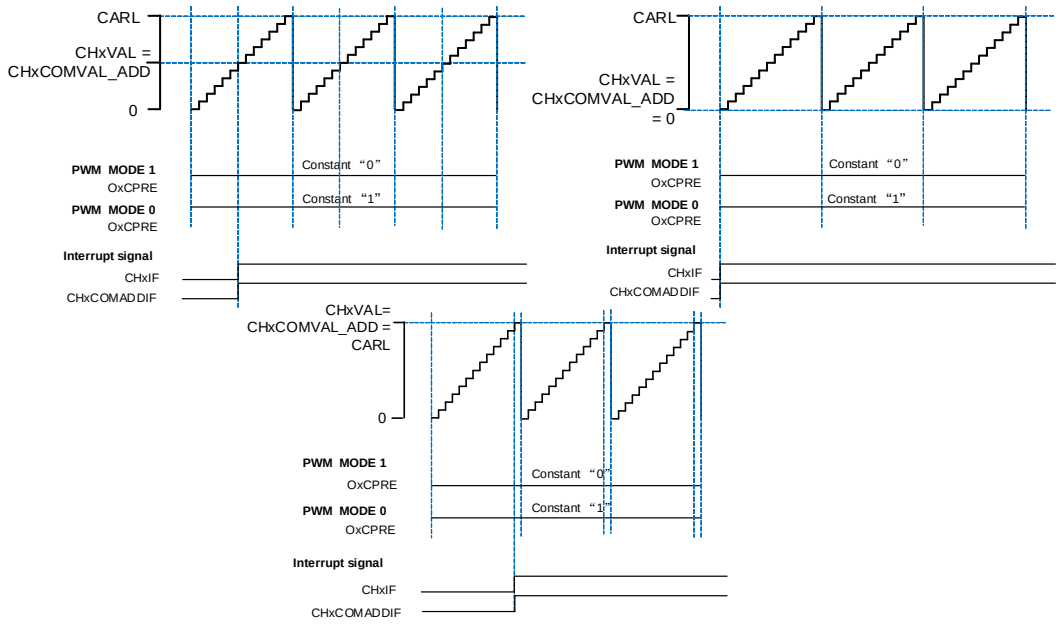
5) CHxVAL < CHxCOMVAL_ADD, and the values of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-63. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD)**



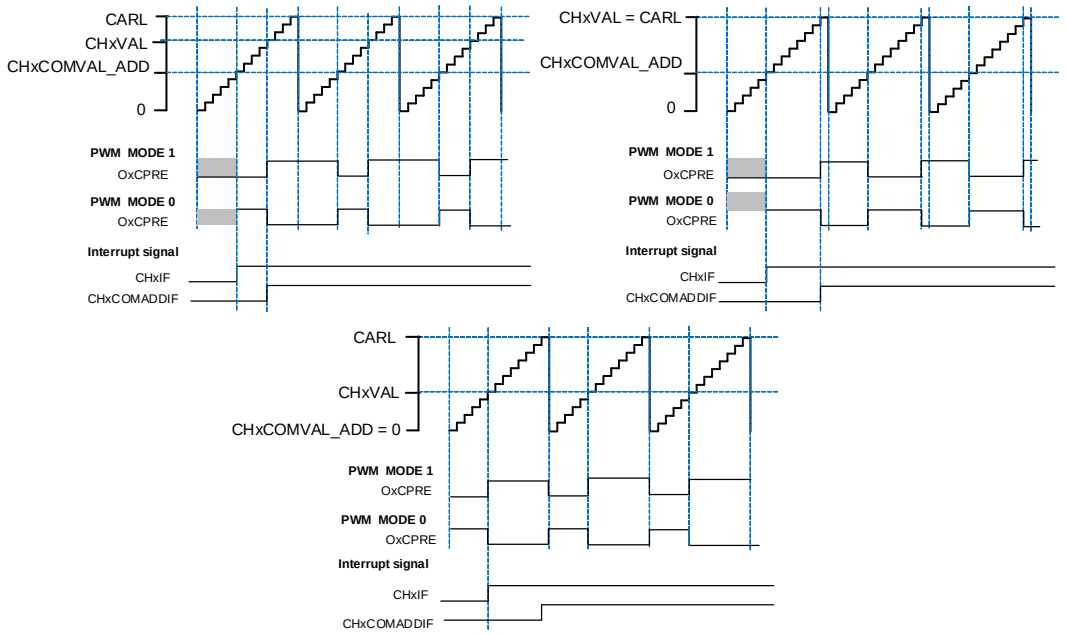

6) CHxVAL = CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-64. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD)**



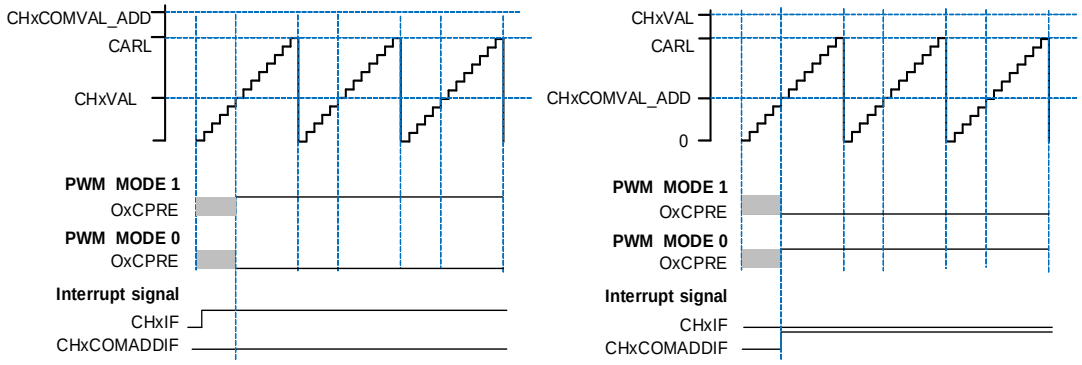

7) CHxVAL > CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-65. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD)**




8) One of the value of CHxVAL and CHxCOMVAL_ADD exceeds CARL.

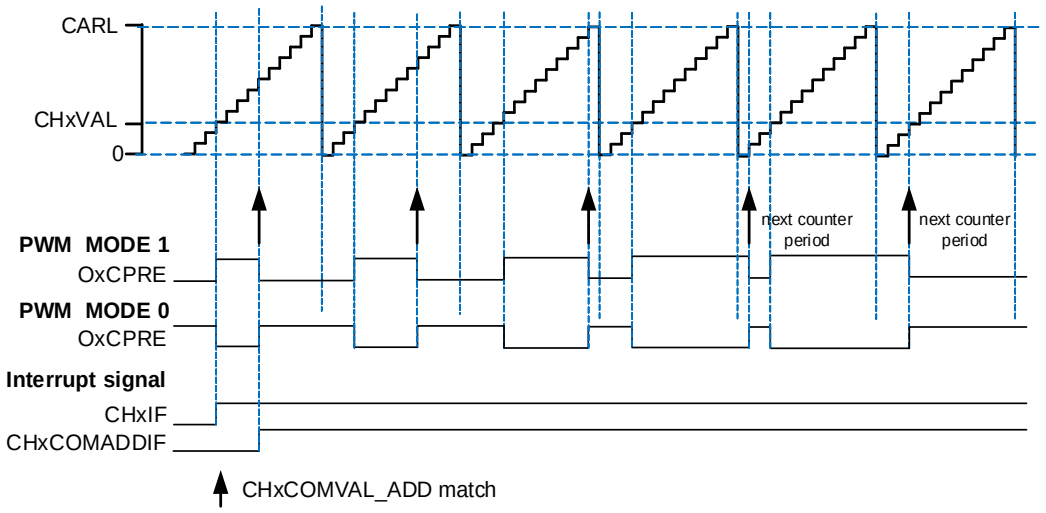
**Figure 24-66. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL**




The composite PWM mode is intended to support the generation of PWM signals where the period is not modified while the signal is being generated, but the duty cycle will be varied. ***Figure 24-67. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD*** shows the PWM output and interrupts waveform.

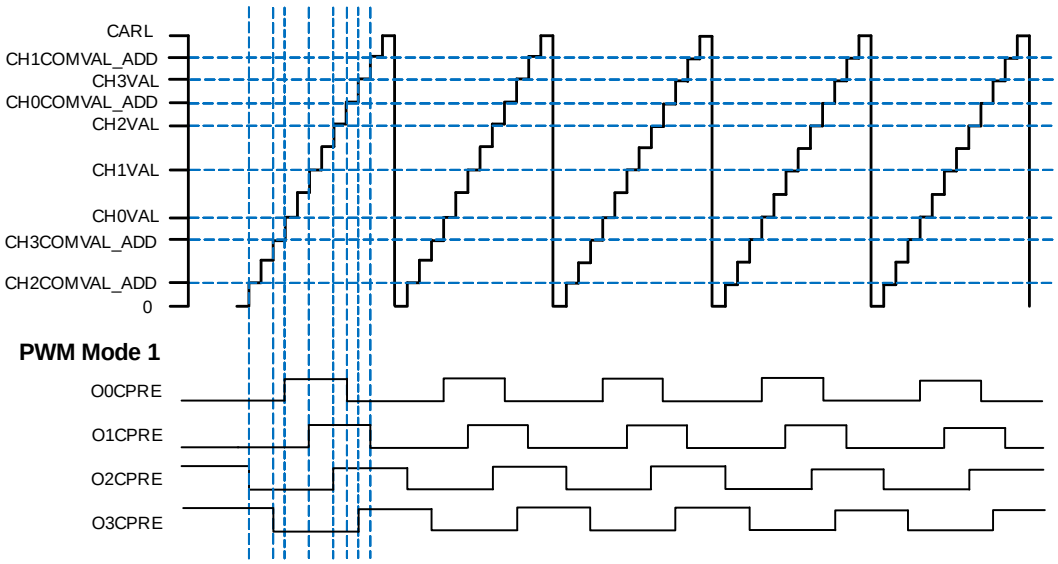
In some cases, the CHxCOMVAL_ADD match can happen on the next counter period (the value of CHxCOMVAL_ADD was written after the counter reaches the value of CHxVAL, and the value of CHxCOMVAL_ADD was less than or equal to the CHxVAL).

**Figure 24-67. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD**




If more than one channels are configured in composite PWM mode, it is possible to fix an offset for the channel x match edge of each pair with respect to other channels. This behavior is useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other pair to help eliminate noise generation. The CHxVAL register value is the shift of the PWM pulse with respect to the beginning of counter period.

**Figure 24-68. Four Channels outputs in Composite PWM mode**




## Output match pulse select

Basing on that CHx_O(x=0..3) outputs are configured by CHxCOMCTL[3:0](x=0..3) bits when the match events occur, the output signal is configured by CHxOMPSEL[1:0](x=0..3) bit to be normal or a pulse.

When the match events occur, the CHxOMPSEL[1:0](x=0..3) bits are used to select the output of OxCPRE which drives CHx_O:

- CHxOMPSEL = 2'b00, the OxCPRE signal is output normally with the configuration of

CHxCOMCTL[3:0] bits;

- CHxOMPSEL = 2'b01, only the counter is counting up, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.
- CHxOMPSEL = 2'b10, only the counter is counting down, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.
- CHxOMPSEL = 2'b11, both the counter is counting up and counting down, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.

**Figure 24-69. CHx_O output with a pulse in edge-aligned mode (CHxOMPSEL≠2'b00)**

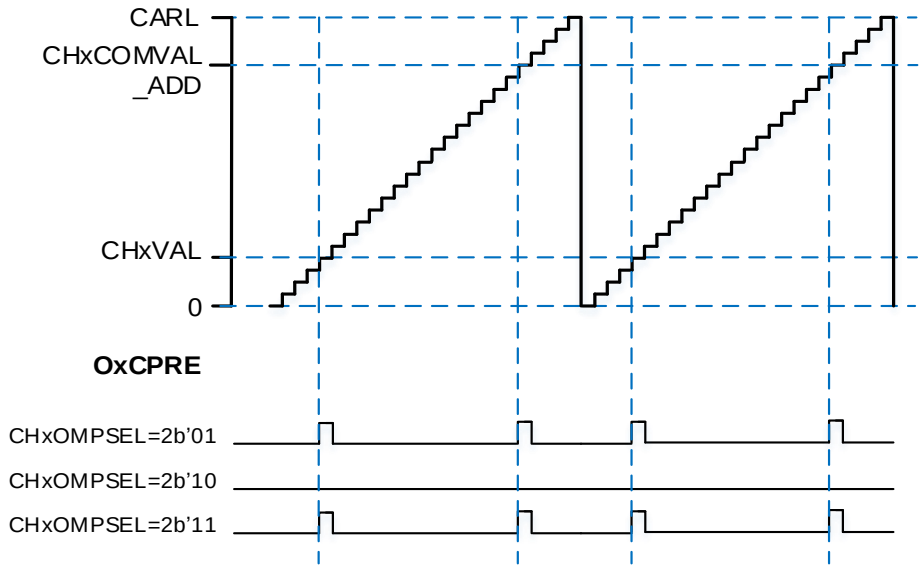
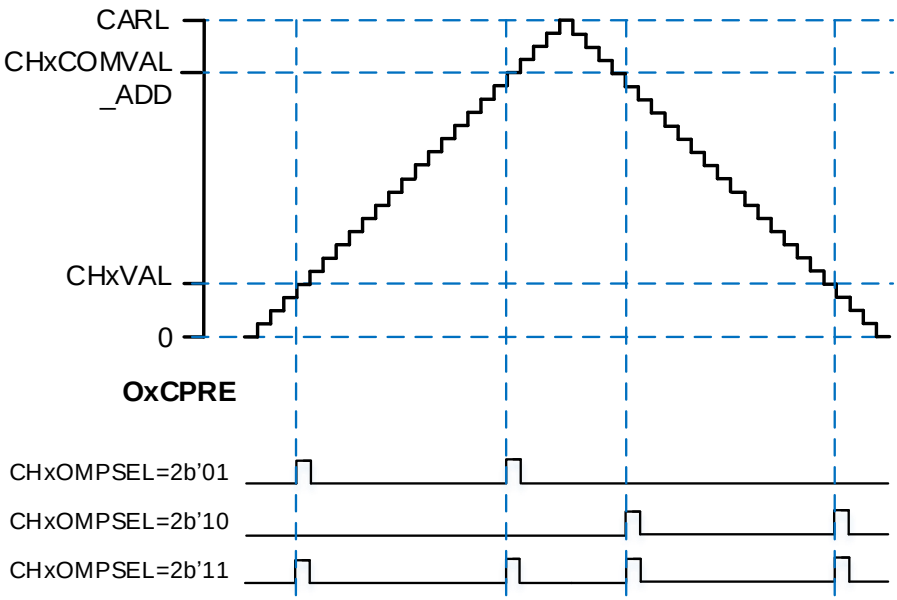



**Figure 24-70. CHx_O output with a pulse in center-aligned mode (CHxOMPSEL≠2'b00)**




### Channel output prepare signal

As is shown in ***Figure 24-59. Output compare logic (x=0,1,2,3)***, when TIMERx is configured in compare match output mode, a middle signal which is OxCPRE signal (Channel x output

prepare signal) will be generated before the channel outputs signal. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit.

The OxCPRE signal has several types of output function. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit. These include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is a forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMERx_CH0 and TIMERx_CH1 pins respectively to interact with each other to generate the counter value.

Setting TSCFGy[4:0](y=0..2) != 5'b00000 to select that the counting direction of timer is determined only by the CI0, only by the CI1, or by the CI0 and the CI1. The DIR bit is modified by hardware automatically during the voltage level change of each direction selection source. The mechanism of changing the counter direction is shown in ***Table 24-11. Counting direction in different quadrature decoder signals***. The CI0FE0 and CI1FE1 are the signals of the CI0 and CI1 after the filtering and polarity selection. The quadrature decoder can be regarded as an external clock with a direction selection. This means that the counter counts continuously from 0 to the counter-reload value. Therefore, users must configure the TIMERx_CAR register before the counter starts to count.

**Table 24-11. Counting direction in different quadrature decoder signals**

| Counting mode | Level | CI0FE0 | | CI1FE1 | |
|---|---|---|---|---|---|
| | | Rising | Falling | Rising | Falling |
| Quadrature decoder mode 0 TSCFG0[4:0]!= 5'b00000 | CI1FE1=1 | Down | Up | - | - |
| | CI1FE1=0 | Up | Down | - | - |
| Quadrature decoder mode 1 TSCFG1[4:0]!= 5'b00000 | CI0FE0=1 | - | - | Up | Down |
| | CI0FE0=0 | - | - | Down | Up |
| Quadrature decoder mode 2 TSCFG2[4:0]!= 5'b00000 | CI1FE1=1 | Down | Up | X | X |
| | CI1FE1=0 | Up | Down | X | X |
| | CI0FE0=1 | X | X | Up | Down |
| | CI0FE0=0 | X | X | Down | Up |

**Note:** "-" means "no counting"; "X" means impossible. "0" means "low level", "1" means "high level".

**Figure 24-71. Example of counter operation in decoder interface mode**

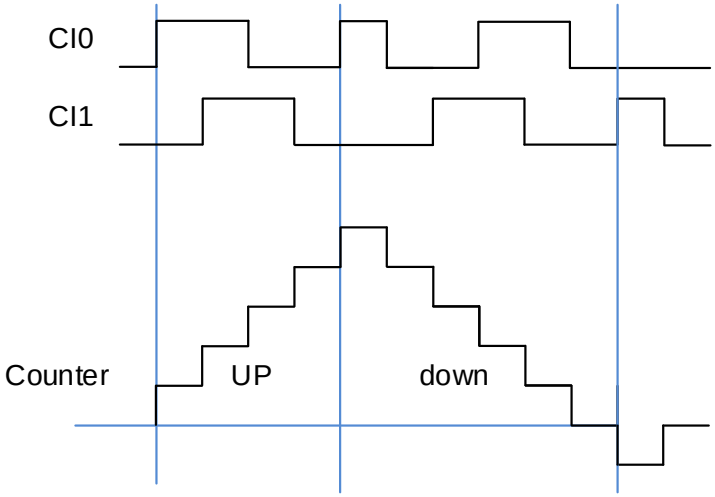
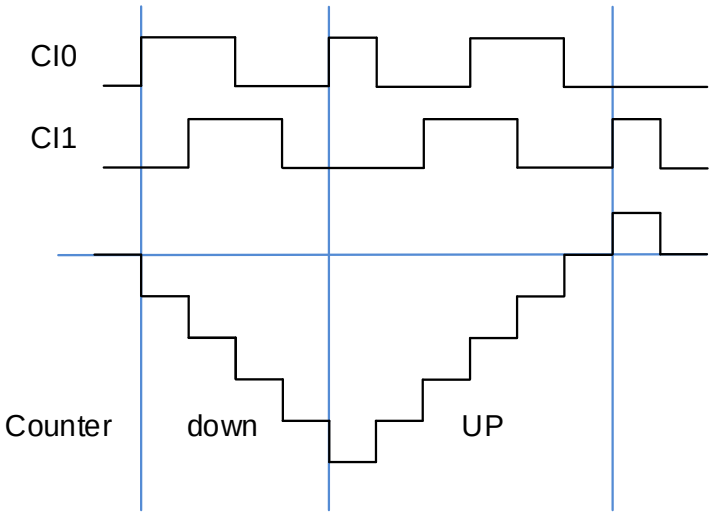



**Figure 24-72. Example of decoder interface mode with CI0FE0 polarity inverted**




### Quadrature decoder signal disconnection detection

The quadrature decoder signal jump detection function can be enabled by setting the DECJDEN bit (in TIMERx_CTL2register) to 1, which can be used to detect whether the level

jump edges (rising or falling) of the CI0 and CI1 signals occur at the same time.

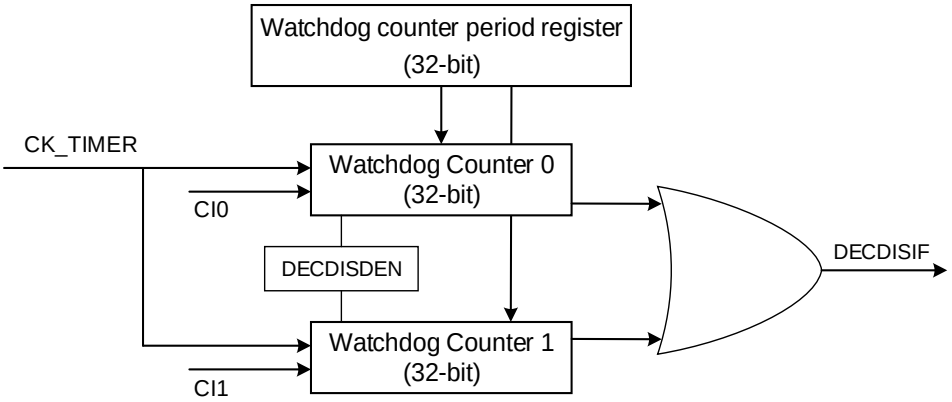
When DECJDEN =1, if the level transitions of the two quadrature signals CI0 and CI1 occur simultaneously, the interrupt flag DECJIF is set, if DECJIE=1, the corresponding interrupt is generated.

The quadrature decoder signal disconnection detection function can be enabled by setting the DECDISDEN bit (in TIMERx_CTL2 register) to 1, which can be used to detect the signal conditions of the CI0 and CI1 signals.

As shown in ***Figure 24-73. Quadrature decoder signal disconnection detection block diagram***. The signal detection module includes two 32-bit watchdog counters and a period register. The CI0 and CI1 signals are used to reset the two watchdog counters respectively.

When DECDISDEN=1, two watchdog counters start counting up at the same time. If the counter continues to count to the watchdog period value (this value is determined by the WDGPER[31:0] bit-field in the TIMERx_WDGPER register), the counter is timeout and the interrupt flag DECDISIF is set. If DECDISIE=1, the corresponding interrupt is generated.

**Figure 24-73. Quadrature decoder signal disconnection detection block diagram**



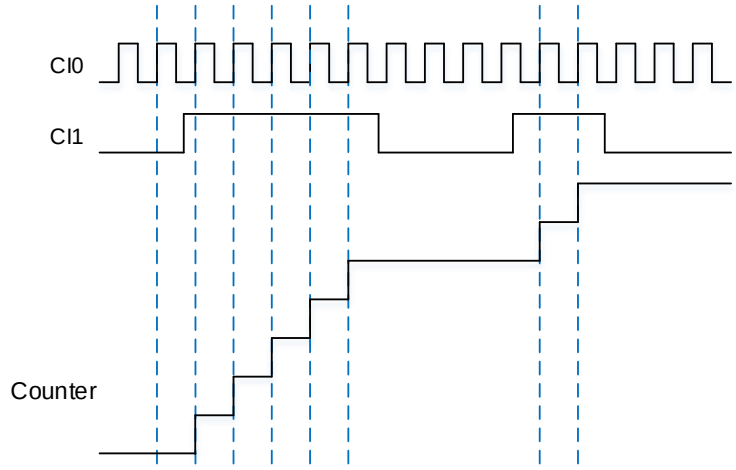

## Non-quadrature decoder

The non-quadrature decoder function has two modes: non-quadrature decoder mode 0 and non-quadrature decoder mode 1, which can be selected by setting TSCFGy[4:0](y=8, 9) != 5'b00000. There are two input sources in these two modes: CI0 and CI1.

When the non-quadrature decoder mode 0 is enabled, the CI0 signal is used as the count pulse and CI1 is used as the count selection signal. When CH1P=0, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is high; When CH1P=1, the counter will count up on the rising edge of the CI0 input signal merely in the case that the CI1 signal is low. The more details is shown in ***Figure 24-74. Example of counter operation in non-quadrature decoder mode 0 with CH1P=0***.

**Figure 24-74. Example of counter operation in non-quadrature decoder mode 0 with**

**CH1P=0**

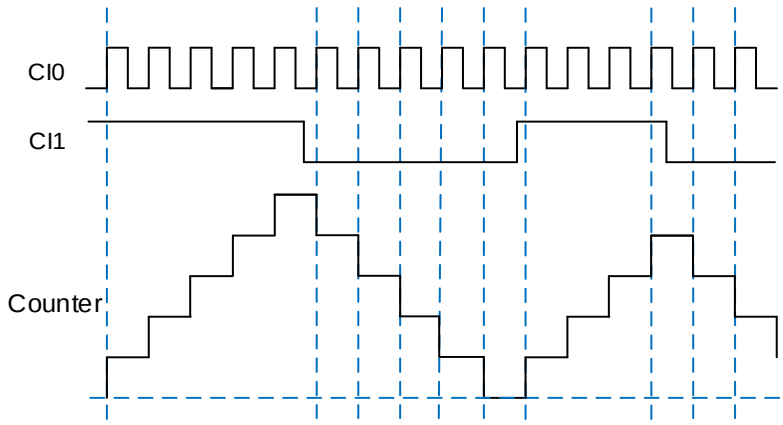



When the non-quadrature decoder mode 1 is enabled, the CI0 signal is used as the count pulse (with the CH0P is used to select the counter edge) and the CI1 signal is used as the count direction selection. The more details is shown in ***Table 24-12. the counter operation in in non-quadrature decoder mode 1*** and ***Figure 24-75. Example of counter operation in non-quadrature decoder mode 1 with CH0P=0***.

**Table 24-12. the counter operation in in non-quadrature decoder mode 1**

| CH0P | level | counter operation |
|---|---|---|
| 0 | CI1 is high | the counter will count up on the rising edge of the CI0 input signal |
| | CI1 is low | the counter will count down on the rising edge of the CI0 input signal |
| 1 | CI1 is high | the counter will count up on the falling edge of the CI0 input signal |
| | CI1 is low | the counter will count down on the falling edge of the CI0 input signal |

**Figure 24-75. Example of counter operation in non-quadrature decoder mode 1 with CH0P=0**



### Hall sensor function

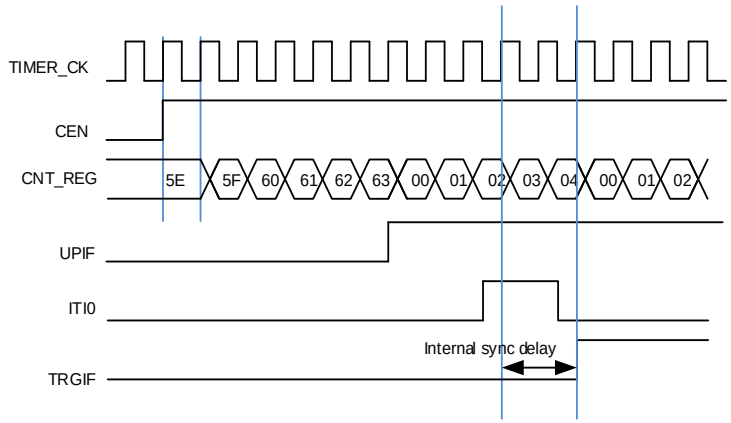
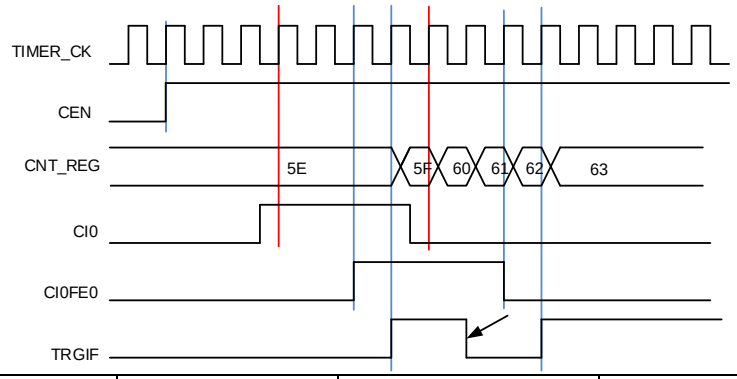
Refer to ***Advanced timer (TIMERx, x=0, 7)Hall sensor function***.

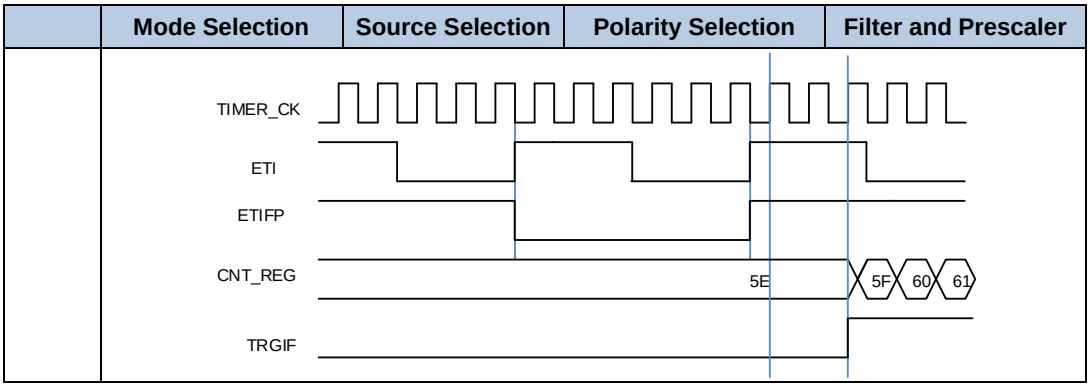
### Master-slave management

The TIMERx can be synchronized with a trigger in several modes including restart mode, pause mode and event mode and so on, which is selected by the TSCFGy[4:0] (y=3..7) in SYSCFG_TIMERxCFG(x=1..4,22,23,30,31).

**Table 24-13. Examples of slave mode**

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| **LIST** | TSCFGy[4:0]<br>y=3: restart mode<br>y=4: pause mode<br>y=5: event mode<br>y=6: external clock mode 0<br>y=7: restart + event mode | TSCFGy[4:0]<br>00000:Mode disable<br>00001: ITI0<br>00010: ITI1<br>00011: ITI2<br>00100: ITI3<br>00101: CI0F_ED<br>00110: CI0FE0<br>00111: CI1FE1<br>01000: ETIFP(1)<br>01001: ITI4<br>01010: ITI5<br>01100: ITI7<br>01110: ITI9<br>01111: ITI10<br>10000: ITI11<br>10001: ITI12<br>10010: ITI13<br>10011: ITI14 | If CI0FE0 or CI1FE1 is selected as the trigger source, configure the CHxP and CHxNP for the polarity selection and inversion.<br>If ETIFP (the filtered output of external trigger input ETI) is selected as the trigger source, configure the ETP for polarity selection and inversion. | For the ITIx, no filter and prescaler can be used.<br>For the CIx, filter can be used by configuring CHxCAPFLT, no prescaler can be used.<br>For the ETIFP, filter can be used by configuring ETFC and prescaler can be used by configuring ETPSC. |
| **Exam1** | **Restart mode**<br>The counter will be cleared and restart when a rising edge of trigger input comes. | TSCFG3[4:0]<br>5'b00001,<br>ITI0 is selected. | For ITI0, no polarity selector can be used. | For the ITI0, no filter and prescaler can be used. |

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| | **Figure 24-76. Restart mode**<br>  <br> | | | |
| **Exam2** | **Pause mode**<br>The counter will be paused when the trigger input is low, and it will start when the trigger input is high. | TSCFG4[4:0] =5'b00110, CI0FE0 is selected. | TI0S = 0 (Non-xor) [CH0NP=0, CH0P=0] CI0FE0 does not invert. The capture event will occur on the rising edge only. | Filter is bypassed in this example. |
| | **Figure 24-77. Pause mode**<br>  <br> | | | |
| **Exam3** | **Event mode**<br>The counter will start to count when a rising edge of trigger input comes. | TSCFG5[4:0] =5'b01000, ETIFP is selected. | ETP = 0, the polarity of ETI does not change. | ETPSC = 1, ETI is divided by 2.<br>ETFC = 0, ETI does not filter. |
| | **Figure 24-78. Event mode** | | | |




(1) The ETI signal can be input from an external ETI pin or provide by on-chip peripherals, please refer to ***Trigger selection for TIMER1_ETI register (TRIGSEL_TIMER1ETI)*** for more details.

### Single pulse mode

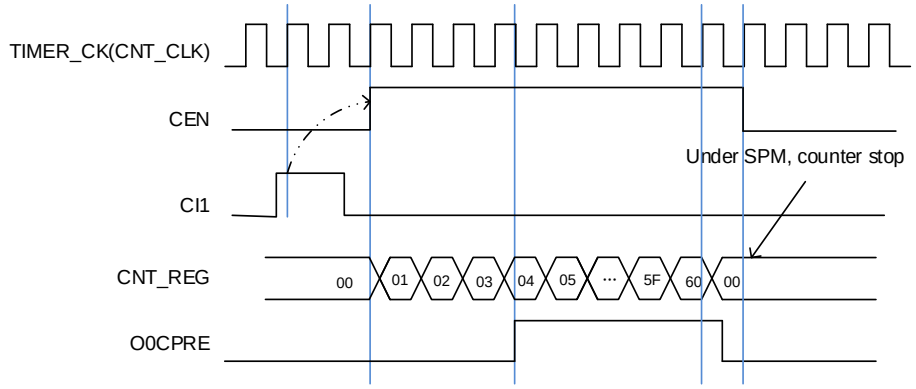
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. If SPM is set, the counter will be cleared and stopped automatically when the next update event occurs. In order to get a pulse waveform, the TIMERx is configured to PWM mode or compare mode by CHxCOMCTL bit.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. Setting the CEN bit to 1 or a trigger signal edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 by software, the counter will be stopped and its value will be held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the active edge of trigger which sets the CEN bit to 1 will enable the counter. However, there exists several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account.

Single pulse mode is also applicable to composite PWM mode (CHxCPWMEN = 1’b1 and CHxMS[2:0] = 3’b000).

**Figure 24-79. Single pulse mode TIMERx_CHxCV = 0x04, TIMERx_CAR=0x60**




## Delayable single pulse mode

Delayable single pulse mode is enabled by setting CHxCOMCTL[3:0] in TIMERx_CHCTLx registers. In this mode, the pulse width of OxCPRE signal is determined by the TIMERx_CAR register.

Once the timer is set to the delayable single pulse mode, the following configuration is required:

- TIMERx need to work in slave mode and TSCFG7[4:0] != 5'b00000 in SYSCFG_TIMERxCFG(x=1..4,22,23,30,31);
- The CHxCOMCTL[3:0] bit-field is setting to 4'b1000 (delayable single pulse mode 0) or 4'b1001 (delayable single pulse mode 1).

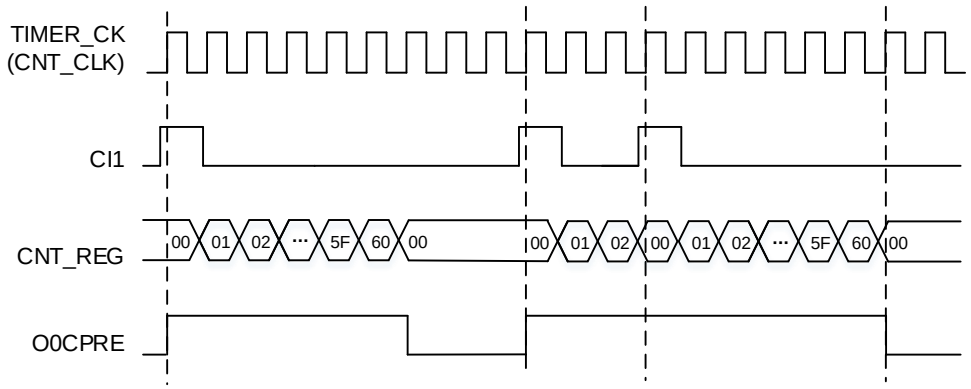
In delayable SPM mode 0. The behavior of OxCPREis performed as in PWM mode 0. When counting up, the OxCPRE is active. When a trigger event occurs, the OxCPRE is inactive. The OxCPRE is active again at the next update event; When counting down, the OxCPRE is inactive, when a trigger event occurs, the OxCPRE is active. The OxCPRE is inactive again at the next update event.

In delayable mode 1. The behavior of OxCPRE is performed as in PWM mode 1. When counting up, the OxCPRE is inactive, when a trigger event occurs, the OxCPRE is active. The OxCPRE is inactive again at the next update event; When counting down, the OxCPRE is active. When a trigger event occurs, the OxCPRE is inactive. The OxCPRE is active again at the next update event.

**Note**:

1) The center-aligned counting mode cannot be used in this mode and the CAM[1:0] = 2'b00( in TIMERx_CTL0 register);
2) When counter counting up (DIR = 0 in TIMERx_CTL0 register), the value of TIMERx_CHxCV should be set to 0; When counting down (DIR =1 in TIMERx_CTL0 register), the value of TIMERx_CHxCV should be greater than or equal to the value of TIMERx_CAR register.

**Figure 24-80. delayable single pulse mode TIMERx_CHxCV=0x00, TIMERx_CAR=0x60**




### Timers interconnection

Please refer to ***Advanced timer (TIMERx, x=0, 7)Timers interconnection***.

### Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. TIMERx will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of TIMERx_DMATB is configured to PADDR (peripheral base address), then DMA will access the TIMERx_DMATB. In fact, TIMERx_DMATB register is only a buffer, timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0 (1 transfer), the timer sends only one DMA request. While if TIMERx_DMATC is not 0, such as 3 (4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8 and DMATA+0xC at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event asserts, (DMATC+1) times request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

### UPIF bit backup

The UPIF bit backup function is enabled by setting UPIFBUEN in the TIMERx_CTL0 register. The UPIF and UPIFBU bits are fully synchronized and without latency.

By using this function, the UPIF bit in the TIMERx_INTF register will be backuped to the UPIFBU bit in the TIMERx_CNT register. This can avoid conflicts when reading the counter and interrupt processing.

### Timer debug mode

When the Cortex®-M7 is halted, and the TIMERx_HOLD configuration bit in DBG_CTL register set to 1, the TIMERx counter stops.

### 24.2.5. Registers definition (TIMERx, x=1,2,3,4,22,23,30,31)

TIMER1 base address: 0x4000 0000

TIMER2 base address: 0x4000 0400

TIMER3 base address: 0x4000 0800

TIMER4 base address: 0x4000 0C00

TIMER22 base address: 0x4000 E000

TIMER23 base address: 0x4000 E400

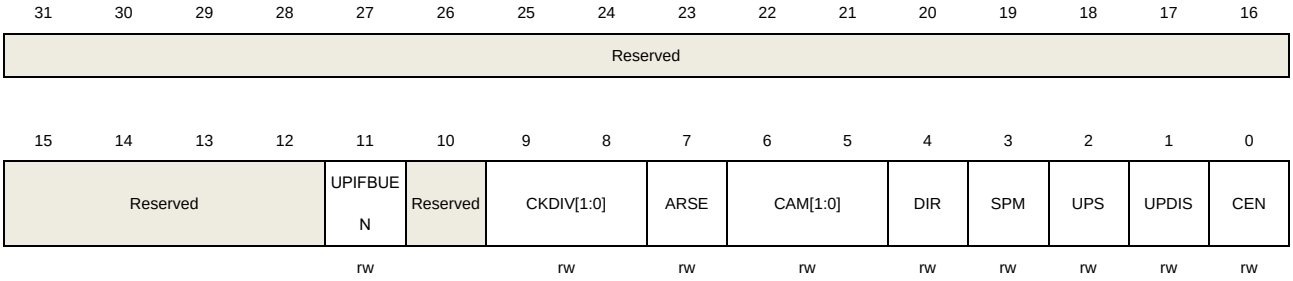
TIMER30 base address: 0x4000 E800

TIMER31 base address: 0x4000 EC00

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | UPIFBUEN | UPIF bit backup enable<br>0: Backup disable. UPIF bit is not backuped to UPIFBU bit in TIMERx_CNT register.<br>1: Backup enabled. UPIF bit is backuped to UPIFBU bit in TIMERx_CNT register. |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division ratio between CK_TIMER (the timer clock) and DTS (the dead time and sampling clock) which is used for the dead time generator and the digital filter.<br>00: $f_{DTS}$= $f_{CK_TIMER}$<br>01: $f_{DTS}$= $f_{CK_TIMER}$ /2<br>10: $f_{DTS}$= $f_{CK_TIMER}$ /4<br>11: Reserved |

| | | |
|---|---|---|
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:5 | CAM[1:0] | Counter align mode selection<br>00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit.<br>01: Center-aligned and counting down assert mode. The counter counts in center-aligned mode and channel is configured in output mode (CHxMS = 3'b000 in TIMERx_CHCTL0 register). Only when the counter is counting down, compare interrupt flag of channels can be set.<br>10: Center-aligned and counting up assert mode. The counter counts in center-aligned mode and channel is configured in output mode (CHxMS = 3'b000 in TIMERx_CHCTL0 register). Only when the counter is counting up, compare interrupt flag of channels can be set.<br>11: Center-aligned and counting up/down assert mode. The counter counts in center-aligned mode and channel is configured in output mode (CHxMS = 3'b000 in TIMERx_CHCTL0 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set.<br>After the counter is enabled, these bits cannot be switched from 0x00 to non 0x00. |
| 4 | DIR | Direction<br>0: Count up<br>1: Count down<br>This bit is read only when the timer is configured in center-aligned mode or decoder mode. |
| 3 | SPM | Single pulse mode<br>0: Single pulse mode is disabled. Counter continues after an update event.<br>1: Single pulse mode is enabled. The CEN bit is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: Any of the following events generates an update interrupt or a DMA request:<br>– The UPG bit is set.<br>– The counter generates an overflow or underflow event.<br>– The slave mode controller generates an update event.<br>1: Only counter overflow/underflow generates an update interrupt or a DMA request. |
| 1 | UPDIS | Update disable<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. The update event is generated and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set.<br>– The counter generates an overflow or underflow event.<br>– The slave mode controller generates an update event. |

| | | |
|---|---|---|
| | | 1: Update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or the slave mode controller generates a hardware reset event. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock mode, pause mode or decoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TI0S | MMC0[2:0] | | | DMAS | Reserved | | |
| | | | | | | | | rw | rw | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | TI0S | Channel 0 trigger input selection<br>0: The TIMERx_CH0 pin input is selected as channel 0 trigger input.<br>1: The result of combinational XOR of TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 pins is selected as channel 0 trigger input. |
| 6:4 | MMC0[2:0] | Master mode control 0<br>These bits control the selection of TRGO0 signal, which is sent by master timer to slave timer for synchronization function.<br>000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO0 pulse occurs. And in the latter case, the signal on TRGO0 is delayed compared to the actual reset.<br>001: Enable. This mode is used to start several timers at the same time or control a slave timer to be enabled in a period. In this mode, the master mode controller selects the counter enable signal as TRGO0. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO0 output, except if the master-slave mode is selected.<br>010: Update. In this mode, the master mode controller selects the update event as TRGO0. |

| | | |
|---|---|---|
| | | 011: Capture/compare pulse. In this mode, the master mode controller generates a TRGO0 pulse when a capture or a compare match occurs in channel 0.<br>100: Compare. In this mode, the master mode controller selects the O0CPRE signal as TRGO0.<br>101: Compare. In this mode, the master mode controller selects the O1CPRE signal as TRGO0.<br>110: Compare. In this mode, the master mode controller selects the O2CPRE signal as TRGO0.<br>111: Compare. In this mode, the master mode controller selects the O3CPRE signal as TRGO0. |
| 3 | DMAS | DMA request source selection<br>0: DMA request of CHx is sent when capture/compare event occurs.<br>1: DMA request of channel CHx is sent when update event occurs. |
| 2:0 | Reserved | Must be kept at reset value. |

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETP | SMC1 | ETPSC[1:0] | | ETFC[3:0] | | | | MSM | Reserved | | | | | | |
| rw | rw | rw | | rw | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | ETP | External trigger polarity<br>This bit specifies the polarity of ETI signal.<br>0: ETI is active at high level or rising edge.<br>1: ETI is active at low level or falling edge. |
| 14 | SMC1 | Part of slave mode controller is used to enable external clock mode 1<br>In external clock mode 1, the counter is clocked by any active edge of the ETIFP signal.<br>0: External clock mode 1 disabled<br>1: External clock mode 1 enabled<br>It is possible to simultaneously use external clock mode 1 with the restart mode, pause mode or event mode. But the TSCFGy[4:0](y=3,4,5) bits must not be 5b'01000 in this case. |

| | | |
|---|---|---|
| | | The external clock input will be ETIFP if external clock mode 0 and external clock mode 1 are enabled at the same time.<br>**Note:** External clock mode 0 enable is in TSCFG6[4:0] bit-field in SYSCFG_TIMERxCFG1 register. |
| 13:12 | ETPSC[1:0] | External trigger prescaler<br>The frequency of external trigger signal ETIFP must not be higher than 1/4 of TIMER_CK frequency. When the frequency of external trigger signal is high, the prescaler can be enabled to reduce ETIFP frequency.<br>00: Prescaler disabled<br>01: ETIFP frequency divided by 2<br>10: ETIFP frequency divided by 4<br>11: ETIFP frequency divided by 8 |
| 11:8 | ETFC[3:0] | External trigger filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETIFP signal and the length of the digital filter applied to ETIFP.<br>0000: Filter disabled. $f_{SAMP}$= $f_{DTS}$, N=1.<br>0001: $f_{SAMP}$= $f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}$= $f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}$= $f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}$=$f_{DTS}$/2, N=6.<br>0101: $f_{SAMP}$=$f_{DTS}$/2, N=8.<br>0110: $f_{SAMP}$=$f_{DTS}$/4, N=6.<br>0111: $f_{SAMP}$=$f_{DTS}$/4, N=8.<br>1000: $f_{SAMP}$=$f_{DTS}$/8, N=6.<br>1001: $f_{SAMP}$=$f_{DTS}$/8, N=8.<br>1010: $f_{SAMP}$=$f_{DTS}$/16, N=5.<br>1011: $f_{SAMP}$=$f_{DTS}$/16, N=6.<br>1100: $f_{SAMP}$=$f_{DTS}$/16, N=8.<br>1101: $f_{SAMP}$=$f_{DTS}$/32, N=5.<br>1110: $f_{SAMP}$=$f_{DTS}$/32, N=6.<br>1111: $f_{SAMP}$=$f_{DTS}$/32, N=8. |
| 7 | MSM | Master-slave mode<br>This bit can be used to synchronize the selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected.<br>0: Master-slave mode disabled<br>1: Master-slave mode enabled |
| 6:0 | Reserved | Must be kept at reset value. |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMADDIE | CH2COMADDIE | CH1COMADDIE | CH0COMADDIE | Reserved | | | | | | | | | | DECDISIE | DECJIE |
| rw | rw | rw | rw | | | | | | | | | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | TRGDEN | Reserved | CH3DEN | CH2DEN | CH1DEN | CH0DEN | UPDEN | Reserved | TRGIE | Reserved | CH3IE | CH2IE | CH1IE | CH0IE | UPIE |
| | rw | | rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3COMADDIE | Channel 3 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 30 | CH2COMADDIE | Channel 2 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 29 | CH1COMADDIE | Channel 1 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 28 | CH0COMADDIE | Channel 0 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 27:18 | Reserved | Must be kept at reset value. |
| 17 | DECDISIE | Quadrature decoder signal disconnection interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for quadrature decoder signal disconnection detection is enabled (when DECDISDEN =1). |
| 16 | DECJIE | Quadrature decoder signal jump (the two signals jump at the same time) interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for quadrature decoder signal jump detection is enabled |

| | | |
|---|---|---|
| | | (when DECJDEN =1). |
| 15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: Disabled<br>1: Enabled |
| 13 | Reserved | Must be kept at reset value. |
| 12 | CH3DEN | Channel 3 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 11 | CH2DEN | Channel 2 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 10 | CH1DEN | Channel 1 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Disabled<br>1: Enabled |
| 7 | Reserved | Must be kept at reset value. |
| 6 | TRGIE | Trigger interrupt enable<br>0: Disabled<br>1: Enabled |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3IE | Channel 3 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 3 | CH2IE | Channel 2 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 2 | CH1IE | Channel 1 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: Disabled |

| | | |
|---|---|---|
| | | 1: Enabled |
| 0 | UPIE | Update interrupt enable<br>0: Disabled<br>1: Enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMADDIF | CH2COMADDIF | CH1COMADDIF | CH0COMADDIF | Reserved | | | | | | | | | | DECDISIF | DECJIF |
| rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | | | | | | | | | rc_w0 | rc_w0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | CH3OF | CH2OF | CH1OF | CH0OF | Reserved | | TRGIF | Reserved | CH3IF | CH2IF | CH1IF | CH0IF | UPIF |
| | | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3COMADDIF | Channel 3 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 30 | CH2COMADDIF | Channel 2 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 29 | CH1COMADDIF | Channel 1 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 28 | CH0COMADDIF | Channel 0 additional compare interrupt flag.<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No channel 0 output compare interrupt occurred<br>1: Channel 0 output compare interrupt occurred<br>**Note:** This flag just used in composite PWM mode (when CH0CPWMEN = 1'b1, CH0MS[2:0] = 3'b000 and CH0COMCTL = 4'b0110 or 4'b0111). |
| 27:18 | Reserved | Must be kept at reset value. |
| 17 | DECDISIF | Quadrature decoder signal disconnection interrupt flag<br>0: No quadrature decoder signal disconnection interrupt occurred<br>1: Quadrature decoder signal disconnection interrupt occurred<br>**Note:** This bit just used for quadrature decoder signal disconnection detection is enabled (when DECDISDEN =1). |
| 16 | DECJIF | Quadrature decoder signal jump (the two signals jump at the same time) interrupt flag |

| | | |
|---|---|---|
| | | 0: No quadrature decoder signal jump interrupt occurred<br>1: Quadrature decoder signal jump interrupt occurred<br>**Note:** This bit just used for quadrature decoder signal jump detection is enabled (when DECJDEN =1). |
| 15:13 | Reserved | Must be kept at reset value. |
| 12 | CH3OF | Channel 3 over capture flag<br>Refer to CH0OF description |
| 11 | CH2OF | Channel 2 over capture flag<br>Refer to CH0OF description |
| 10 | CH1OF | Channel 1 over capture flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8:7 | Reserved | Must be kept at reset value. |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set by hardware on trigger event and cleared by software.<br>When the slave mode controller is enabled in all modes but pause mode, an active edge of trigger input generates a trigger event. When the slave mode controller is enabled in pause mode, either edge of the trigger input can generate a trigger event.<br>0: No trigger event occurred<br>1: Trigger interrupt occurred |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3IF | Channel 3 capture/compare interrupt flag<br>Refer to CH0IF description |
| 3 | CH2IF | Channel 2 capture/compare interrupt flag<br>Refer to CH0IF description |
| 2 | CH1IF | Channel 1 capture/compare interrupt flag<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 capture/compare interrupt flag<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in input mode, this flag is set when a capture event occurs. If channel 0 is in output mode, this flag is set when a compare event occurs.<br>If channel 0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV.<br>0: No channel 0 interrupt occurred |

| | | |
|---|---|---|
| | | 1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware when an update event occurs and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMADDG | CH2COMADDG | CH1COMADDG | CH0COMADDG | Reserved | | | | | | | | | | | |
| w | w | w | w | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | TRGG | Reserved | CH3G | CH2G | CH1G | CH0G | UPG |
| | | | | | | | | | w | | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3COMADDG | Channel 3 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 30 | CH2COMADDG | Channel 2 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 29 | CH1COMADDG | Channel 1 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 28 | CH0COMADDG | Channel 0 additional compare event generation.<br>This bit is set by software to generate a compare event in channel 0 additional, it is automatically cleared by hardware.<br>When this bit is set, the CH0COMADDIF flag will be set, and the corresponding interrupt will be sent if enabled.<br>0: No generate a channel 0 additional compare event<br>1: Generate a channel 0 additional compare event<br>**Note:** This bit just used in composite PWM mode(when CH0CPWMEN=1). |
| 27:7 | Reserved | Must be kept at reset value. |
| 6 | TRGG | Trigger event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register will be set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a trigger event |

| | | |
|---|---|---|
| | | 1: Generate a trigger event |
| 5 | Reserved | Must be kept at reset value. |
| 4 | CH3G | Channel 3 capture or compare event generation<br>Refer to CH0G description |
| 3 | CH2G | Channel 2 capture or compare event generation<br>Refer to CH0G description |
| 2 | CH1G | Channel 1 capture or compare event generation<br>Refer to CH0G description |
| 1 | CH0G | Channel 0 capture or compare event generation<br>This bit is set by software to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag has been set.<br>0: No generate a channel 0 capture or compare event<br>1: Generate a channel 0 capture or compare event |
| 0 | UPG | Update event generation<br>This bit can be set by software, and automatically cleared by hardware. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, while in down counting mode it takes the auto-reload value. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1MS[2] | CH0MS[2] | CH1COMADDSEN | CH0COMADDSEN | Reserved | | | CH1COMCTL[3] | Reserved | | | | | | | CH0COMCTL[3] |
| | | Reserved | Reserved | | | | Reserved | | | | | | | | Reserved |
| rw | rw | rw | rw | | | | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1COMCEN | CH1COMCTL[2:0] | | | CH1COMSEN | Reserved | CH1MS[1:0] | | CH0COMCEN | CH0COMCTL[2:0] | | | CH0COMSEN | Reserved | CH0MS[1:0] | |
| CH1CAPFLT[3:0] | | | | CH1CAPPSC[1:0] | | | | CH0CAPFLT[3:0] | | | | CH0CAPPSC[1:0] | | | |
| rw | | | | rw | | rw | | rw | | | | rw | | rw | |

**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH1MS[2] | Channel 1 I/O mode selection<br>Refer to CH1MS[1:0]description |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Refer to CH0MS[1:0] description |
| 29 | CH1COMADDSEN | Channel 1 additional compare output shadow enable<br>Refer to CH0COMADDSEN description. |
| 28 | CH0COMADDSEN | Channel 0 additional compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0COMV_ADD register which updates at each update event will be enabled.<br>0: Channel 0 additional compare output shadow disabled<br>1: Channel 0 additional compare output shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | CH1COMCTL[3] | Channel 1 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | CH0COMCTL[3] | Channel 0 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 15 | CH1COMCEN | Channel 1 output compare clear enable<br>Refer to CH0COMCEN description |
| 14:12 | CH1COMCTL[2:0] | Channel 1 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 11 | CH1COMSEN | Channel 1 output compare shadow enable<br>Refer to CH0COMSEN description |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. The CH1MS[2:0] bit-field is writable only when the channel is not active (the CH1EN bit in TIMERx_CHCTL2 register is reset).<br>000: Channel 1 is configured as output.<br>001: Channel 1 is configured as input, IS1 is connected to CI1FE1.<br>010: Channel 1 is configured as input, IS1 is connected to CI0FE1.<br>011: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=1...4,22,23,30,31) register).<br>100~111: Reserved. |

| | | |
|---|---|---|
| 7 | CH0COMCEN | Channel 0 output compare clear enable<br>When this bit is set, the O0CPRE signal is cleared when high level is detected on ETIFP input.<br>0: Channel 0 output compare clear disabled<br>1: Channel 0 output compare clear enabled |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>The CH0COMCTL[3] and CH0COMCTL[2:0] bit-field control the behavior of O0CPRE which drives CH0_O. The active level of O0CPRE is high, while the active level of CH0_O depends on CH0P bit.<br>0000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>0010: Clear the channel output on match. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.<br>0011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.<br>0100: Force low. O0CPRE is forced low level.<br>0101: Force high. O0CPRE is forced high level.<br>0110: PWM mode 0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV, otherwise it is inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV, otherwise it is active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of O0CPRE is performed as in PWM mode 0. When counting up, the O0CPRE is active. When a trigger event occurs, the O0CPRE is inactive. The O0CPRE is active again at the next update event; When counting down, the O0CPRE is inactive, when a trigger event occurs, the O0CPRE is active. The O0CPRE is inactive again at the next update event.<br>1001: Delayable SPM mode 1. The behavior of O0CPRE is performed as in PWM mode 1. When counting up, the O0CPRE is inactive, when a trigger event occurs, the O0CPRE is active. The O0CPRE is inactive again at the next update event; When counting down, the O0CPRE is active. When a trigger event occurs, the O0CPRE is inactive. The O0CPRE is active again at the next update event.<br>1010~1111: Reserved.<br>**Note**: In the composite PWM mode (CH0CPWMEN = 1'b1 and CH0MS = 3'b000), the PWM signal output in channel 0 is composited by TIMERx_CH0CV and TIMERx_CH0COMV_ADD. Please refer to ***Composite PWM mode*** for more details.<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from "Timing" mode to "PWM" mode or the result of the |

| | | |
|---|---|---|
| | | comparison changes. |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register which updates at each update event will be enabled.<br>0: Channel 0 output compare shadow disabled<br>1: Channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. The CH0MS[2:0] bit-field is writable only when the channel is not active (the CH0EN bit in TIMERx_CHCTL2 register is reset).<br>000: Channel 0 is configured as output.<br>001: Channel 0 is configured as input, IS0 is connected to CI0FE0.<br>010: Channel 0 is configured as input, IS0 is connected to CI1FE0.<br>011: Channel 0 is configured as input, IS0 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=1...4,22,23,30,31) register).<br>100~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH1MS[2] | Channel 1 I/O mode selection<br>Same as output compare mode. |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Same as output compare mode. |
| 29:16 | Reserved | Must be kept at reset value. |
| 15:12 | CH1CAPFLT[3:0] | Channel 1 input capture filter control<br>Refer to CH0CAPFLT description. |
| 11:10 | CH1CAPPSC[1:0] | Channel 1 input capture prescaler<br>Refer to CH0CAPPSC description. |
| 9:8 | CH1MS[1:0] | Channel 1 I/O mode selection<br>Same as output compare mode. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2. |

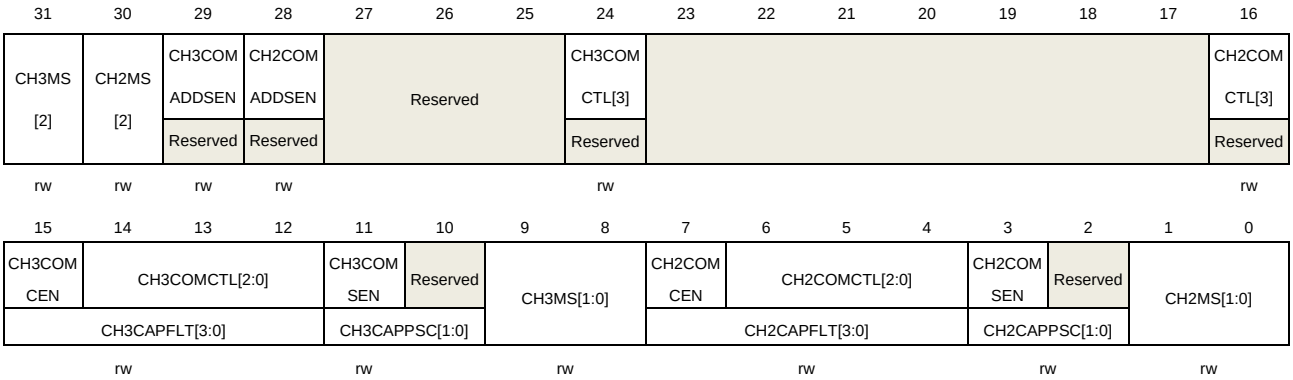
0010: $f_{SAMP}$= $f_{CK_TIMER}$, N=4.
0011: $f_{SAMP}$= $f_{CK_TIMER}$, N=8.
0100: $f_{SAMP}$=$f_{DTS}$/2, N=6.
0101: $f_{SAMP}$=$f_{DTS}$/2, N=8.
0110: $f_{SAMP}$=$f_{DTS}$/4, N=6.
0111: $f_{SAMP}$=$f_{DTS}$/4, N=8.
1000: $f_{SAMP}$=$f_{DTS}$/8, N=6.
1001: $f_{SAMP}$=$f_{DTS}$/8, N=8.
1010: $f_{SAMP}$=$f_{DTS}$/16, N=5.
1011: $f_{SAMP}$=$f_{DTS}$/16, N=6.
1100: $f_{SAMP}$=$f_{DTS}$/16, N=8.
1101: $f_{SAMP}$=$f_{DTS}$/32, N=5.
1110: $f_{SAMP}$=$f_{DTS}$/32, N=6.
1111: $f_{SAMP}$=$f_{DTS}$/32, N=8.

| | | |
|---|---|---|
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>Same as output compare mode. |

### Channel control register 1 (TIMERx_CHCTL1)

Address offset: 0x1C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3MS[2] | Channel 3 I/O mode selection |

| | | |
|---|---|---|
| | | Refer to CH3MS[1:0]description. |
| 30 | CH2MS[2] | Channel 2 I/O mode selection<br>Refer to CH2MS[1:0] description. |
| 29 | CH3COMADDSEN | Channel 3 additional compare output shadow enable<br>Refer to CH2COMADDSEN description. |
| 28 | CH2COMADDSEN | Channel 2 additional compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH2COMV_ADD register which updates at each update event will be enabled.<br>0: Channel 2 additional compare shadow disabled<br>1: Channel 2 additional compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | CH3COMCTL[3] | Channel 3 compare output control<br>Refer to CH2COMCTL[2:0] description |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | CH2COMCTL[3] | Channel 2 compare output control<br>Refer to CH2COMCTL[2:0] description |
| 15 | CH3COMCEN | Channel 3 output compare clear enable<br>Refer to CH2COMCEN description |
| 14:12 | CH3COMCTL[2:0] | Channel 3 compare output control<br>Refer to CH2COMCTL[2:0] description |
| 11 | CH3COMSEN | Channel 3 output compare shadow enable<br>Refer to CH2COMSEN description |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CH3MS[1:0] | Channel 3 I/O mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. The CH3MS[2:0] bit-field is writable only when the channel is not active (the CH3EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 3 is configured as output.<br>01: Channel 3 is configured as input, IS3 is connected to CI3FE3.<br>10: Channel 3 is configured as input, IS3 is connected to CI2FE3.<br>11: Channel 3 is configured as input, IS3 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=1...4,22,23,30,31) register).<br>100~111: Reserved. |
| 7 | CH2COMCEN | Channel 2 output compare clear enable.<br>When this bit is set, the O2CPRE signal is cleared when high level is detected on |

| | | |
|---|---|---|
| | | ETIFP input.<br>0: Channel 2 output compare clear disabled<br>1: Channel 2 output compare clear enabled |
| 6:4 | CH2COMCTL[2:0] | Channel 2 compare output control<br>This bit-field controls the behavior of O2CPRE which drives CH2_O. The active level of O2CPRE is high, while the active level of CH2_O depends on CH2P bit.<br>0000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. O2CPRE signal is forced high when the counter matches the output compare register TIMERx_CH2CV.<br>0010: Clear the channel output on match. O2CPRE signal is forced low when the counter matches the output compare register TIMERx_CH2CV.<br>0011: Toggle on match. O2CPRE toggles when the counter matches the output compare register TIMERx_CH2CV.<br>0100: Force low. O2CPRE is forced low level.<br>0101: Force high. O2CPRE is forced high level.<br>0110: PWM mode 0. When counting up, O2CPRE is active as long as the counter is smaller than TIMERx_CH2CV, otherwise it is inactive. When counting down, O2CPRE is inactive as long as the counter is larger than TIMERx_CH2CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, O2CPRE is inactive as long as the counter is smaller than TIMERx_CH2CV, otherwise it is active. When counting down, O2CPRE is active as long as the counter is larger than TIMERx_CH2CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of O2CPRE is performed as in PWM mode 0. When counting up, the O2CPRE is active. When a trigger event occurs, the O2CPRE is inactive. The O2CPRE is active again at the next update event; When counting down, the O2CPRE is inactive, when a trigger event occurs, the O2CPRE is active. The O2CPRE is inactive again at the next update event.<br>1001: Delayable SPM mode 1. The behavior of O2CPRE is performed as in PWM mode 1. When counting up, the O2CPRE is inactive, when a trigger event occurs, the O2CPRE is active. The O2CPRE is inactive again at the next update event; When counting down, the O2CPRE is active. When a trigger event occurs, the O2CPRE is inactive. The O2CPRE is active again at the next update event.<br>1010~1111: Reserved.<br>**Note**: In the composite PWM mode (CH2CPWMEN = 1'b1 and CH2MS = 3'b000), the PWM signal output in channel 2 is composited by TIMERx_CH2CV and TIMERx_CH2COMV_ADD. Please refer to ***Composite PWM mode*** for more details.<br>If configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from "Timing" mode to "PWM" mode or the result of the comparison changes. |

| | | |
|---|---|---|
| 3 | CH2COMSEN | Channel 2 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled.<br>0: Channel 2 output compare shadow disabled<br>1: Channel 2 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | CH2MS[1:0] | Channel 2 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. The CH2MS[2:0] bit-field is writable only when the channel is not active (the CH2EN bit in TIMERx_CHCTL2 register is reset).<br>00: Channel 2 is configured as output.<br>01: Channel 2 is configured as input, IS2 is connected to CI2FE2.<br>10: Channel 2 is configured as input, IS2 is connected to CI3FE2.<br>11: Channel 2 is configured as input, IS2 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=1...4,22,23,30,31) register).<br>100~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3MS[2] | Channel 3 I/O mode selection<br>Same as output compare mode. |
| 30 | CH2MS[2] | Channel 2 I/O mode selection<br>Same as output compare mode. |
| 31:16 | Reserved | Must be kept at reset value. |
| 15:12 | CH3CAPFLT[3:0] | Channel 3 input capture filter control<br>Refer to CH2CAPFLT description. |
| 11:10 | CH3CAPPSC[1:0] | Channel 3 input capture prescaler<br>Refer to CH2CAPPSC description. |
| 9:8 | CH3MS[1:0] | Channel 3 mode selection<br>Same as output compare mode. |
| 7:4 | CH2CAPFLT[3:0] | Channel 2 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI2 input signal and the length of the digital filter applied to CI2.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4. |

0011: $f_{SAMP}$= $f_{CK_TIMER}$, N=8.
0100: $f_{SAMP}$=$f_{DTS}$/2, N=6.
0101: $f_{SAMP}$=$f_{DTS}$/2, N=8.
0110: $f_{SAMP}$=$f_{DTS}$/4, N=6.
0111: $f_{SAMP}$=$f_{DTS}$/4, N=8.
1000: $f_{SAMP}$=$f_{DTS}$/8, N=6.
1001: $f_{SAMP}$=$f_{DTS}$/8, N=8.
1010: $f_{SAMP}$=$f_{DTS}$/16, N=5.
1011: $f_{SAMP}$=$f_{DTS}$/16, N=6.
1100: $f_{SAMP}$=$f_{DTS}$/16, N=8.
1101: $f_{SAMP}$=$f_{DTS}$/32, N=5.
1110: $f_{SAMP}$=$f_{DTS}$/32, N=6.
1111: $f_{SAMP}$=$f_{DTS}$/32, N=8.

| | | |
|---|---|---|
| 3:2 | CH2CAPPSC[1:0] | Channel 2 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | CH2MS[1:0] | Channel 2 mode selection<br>Same as output compare mode. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20
Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3NP | Reserved | CH3P | CH3EN | CH2NP | Reserved | CH2P | CH2EN | CH1NP | Reserved | CH1P | CH1EN | CH0NP | Reserved | CH0P | CH0EN |
| rw | | rw | rw | rw | | rw | rw | rw | | rw | rw | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | CH3NP | Channel 3 complementary capture/compare polarity<br>Refer to CH0NP description. |
| 14 | Reserved | Must be kept at reset value. |
| 13 | CH3P | Channel 3 capture/compare function polarity |

| | | |
|---|---|---|
| | | Refer to CH0P description |
| 12 | CH3EN | Channel 3 capture/compare function enable<br>Refer to CH0EN description |
| 11 | CH2NP | Channel 2 complementary capture/compare polarity<br>Refer to CH0NP description. |
| 10 | Reserved | Must be kept at reset value. |
| 9 | CH2P | Channel 2 capture/compare function polarity<br>Refer to CH0P description |
| 8 | CH2EN | Channel 2 capture/compare function enable<br>Refer to CH0EN description |
| 7 | CH1NP | Channel 1 complementary capture/compare polarity<br>Refer to CH0NP description. |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CH1P | Channel 1 capture/compare function polarity<br>Refer to CH0P description |
| 4 | CH1EN | Channel 1 capture/compare function enable<br>Refer to CH0EN description |
| 3 | CH0NP | Channel 0 complementary output polarity<br>When channel 0 complementary is configured in output mode, this bit must be kept at reset value.<br>When CH0 is configured in input mode, in conjunction with CH0P, this bit is used to define the polarity of CH0.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 2 | Reserved | Must be kept at reset value. |
| 1 | CH0P | Channel 0 capture/compare function polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.<br>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.<br>00: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.<br>01: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.<br>10: Reserved. |

| | | |
|---|---|---|
| | | 11: Noninverted/both CIxFE0's edges.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare function enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT) (TIMERx, x= 2,3,30,31)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UPIFBU | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:0 | CNT[15:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

### Counter register (TIMERx_CNT) (TIMERx, x= 1,4,22,23)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[31] | CNT[30:16] | | | | | | | | | | | | | | |
| UPIFBU | | | | | | | | | | | | | | | |
| rw /r | rw | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| CNT[15:0] |
|---|
| rw |

**UPIFBUEN = 0:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CNT[31:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

**UPIFBUEN = 1:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register, and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |
| 30 | CNT[30:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

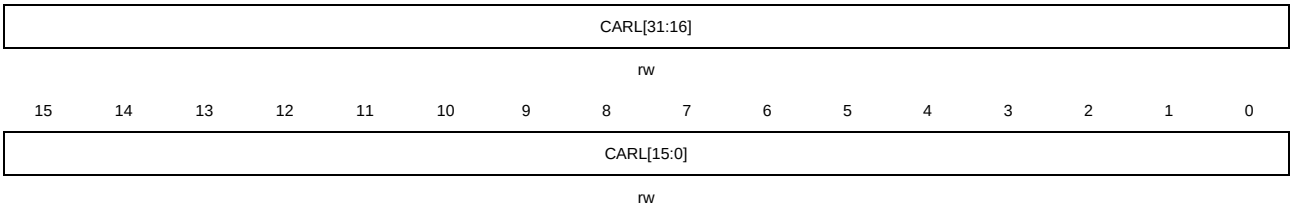
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



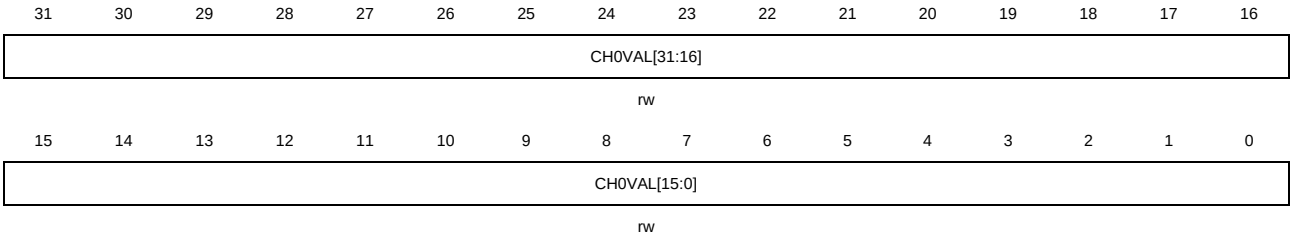

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CARL[31:16] | Counter auto reload value (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-field specifies the auto reload value of the counter. |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



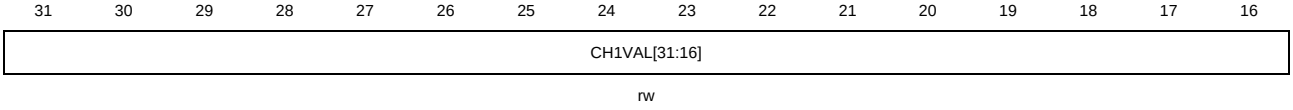

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH0VAL[31:16] | Capture/compare value of channel 0 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH0VAL[15:0] | Capture/compare value of channel 0<br>When channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH1VAL[31:16] | Capture/compare value of channel 1(bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH1VAL[15:0] | Capture/compare value of channel 1<br>When channel 1 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 2 capture/compare value register (TIMERx_CH2CV)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2VAL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH2VAL[31:16] | Capture/compare value of channel 2 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH2VAL[15:0] | Capture/compare value of channel 2<br>When channel 2 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 2 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 3 capture/compare value register (TIMERx_CH3CV)

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| CH3VAL[31:16] | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CH3VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH3VAL[31:16] | Capture/compare value of channel 3 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH3VAL[15:0] | Capture/compare value of channel 3<br>When channel 3 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 0 additional compare value register (TIMERx_CH0COMV_ADD)

Address offset: 0x64

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0COMVAL_ADD[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CH0COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH0COMVAL_ADD [31:16] | Additional compare value of channel 0 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH0COMVAL_ADD [15:0] | Additional compare value of channel 0<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 1 additional compare value register (TIMERx_CH1COMV_ADD)

Address offset: 0x68

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1COMVAL_ADD[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CH1COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH1COMVAL_ADD [31:16] | Additional compare value of channel 1 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH1COMVAL_ADD [15:0] | Additional compare value of channel 1<br>When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 2 additional compare value register (TIMERx_CH2COMV_ADD)

Address offset: 0x6C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH2COMVAL_ADD[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CH2COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH2COMVAL_ADD [31:16] | Additional compare value of channel 2 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH2COMVAL_ADD [15:0] | Additional compare value of channel 2<br>When channel 2 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 3 additional compare value register (TIMERx_CH3COMV_ADD)

Address offset: 0x70

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMVAL_ADD[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CH3COMVAL_ADD [31:16] | Additional compare value of channel 3 (bit 16 to 31)<br>This bit-field only for TIMER1/ 4/ 22/ 23. |
| 15:0 | CH3COMVAL_ADD [15:0] | Additional compare value of channel 3<br>When channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Control register 2 (TIMERx_CTL2)

Address offset: 0x74

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3C PWMEN | CH2C PWMEN | CH1C PWMEN | CH0C PWMEN | Reserved | | | | | | | | DECDISD EN | DECJDEN | Reserved | |
| rw | rw | rw | rw | | | | | | | | | rw | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH3OMPSEL[1:0] | | CH2OMPSEL[1:0] | | CH1OMPSEL[1:0] | | CH0OMPSEL[1:0] | | Reserved | | | | | | | |
| rw | | rw | | rw | | rw | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH3CPWMEN | Channel 3 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 30 | CH2CPWMEN | Channel 2 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 29 | CH1CPWMEN | Channel 1 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 28 | CH0CPWMEN | Channel 0 composite PWM mode enable<br>0: Disabled |

| | | |
|---|---|---|
| | | 1: Enabled |
| 27:20 | Reserved | Must be kept at reset value. |
| 19 | DECDISDEN | Quadrature decoder signal disconnection detection enable<br>0: Quadrature decoder signal disconnection detection is disabled<br>1: Quadrature decoder signal disconnection detection is enabled |
| 18 | DECJDEN | Quadrature decoder signal jump (the two signals jump at the same time) detection enable<br>0: Quadrature decoder signal jump detection is disabled<br>1: Quadrature decoder signal jump detection is enabled |
| 17:16 | Reserved | Must be kept at reset value. |
| 15:14 | CH3OMPSEL[1:0] | Channel 3 output match pulse select<br>When the match events occur, this bit is used to select the output of O3CPRE which drives CH3_O.<br>00: The O3CPRE signal is output normally with the configuration of CH3COMCTL [2:0] bits.<br>01: Only the counter is counting up, the O3CPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.<br>10: Only the counter is counting down, the O3CPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.<br>11: Both the counter is counting up and counting down, the O3CPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle. |
| 13:12 | CH2OMPSEL[1:0] | Channel 2 output match pulse select<br>When the match events occur, this bit is used to select the output of O2CPRE which drives CH2_O.<br>00: The O2CPRE signal is output normal with the configuration of CH2COMCTL [2:0] bits.<br>01: Only when the counter is counting up, the O2CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Only when the counter is counting down, the O2CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>11: Both when the counter is counting up and counting down, the O2CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle. |
| 11:10 | CH1OMPSEL[1:0] | Channel 1 output match pulse select<br>When the match events occur, this bit is used to select the output of O1CPRE which drives CH1_O.<br>00: The O1CPRE signal is output normal with the configuration of CH1COMCTL [2:0] bits.<br>01: Only when the counter is counting up, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle. |

| | | |
|---|---|---|
| | | 10: Only when the counter is counting down, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>11: Both when the counter is counting up and counting down, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle. |
| 9:8 | CH0OMPSEL[1:0] | Channel 0 output match pulse select<br>When the match events occur, this bit is used to select the output of O0CPRE which drives CH0_O.<br>00: The O0CPRE signal is output normal with the configuration of CH0COMCTL [2:0] bits.<br>01: Only when the counter is counting up, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Only when the counter is counting down, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>11: Both when the counter is counting up and counting down, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle. |
| 7:0 | Reserved | Must be kept at reset value. |

### Watchdog counter period register(TIMERx_WDGPER)

Address offset: 0x94
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDGPER[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDGPER[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | WDGPER[31:0] | Watchdog counter period value<br>This register contains the period of the two watchdog counter. When the counters continus to count to this value, the counter will timeout and the interrupt flag DECDISIF is set. If DECDISIE=1, the corresponding interrupt is generated.<br>Note: This register is just used in quadrature decoder signal disconnection detection function(with DECDISDEN =1). |

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0xE0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DMATC[5:0] | | | | | | Reserved | | DMATA[5:0] | | | | | |
| | | rw | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:8 | DMATC[5:0] | DMA transfer count<br>This field defines the times of accessing(R/W) the TIMERx_DMATB register by DMA.<br>6'b000000: transfer 1 time<br>6'b000001: transfer 2 times<br>…<br>6'b100101: transfer 38 times |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:0 | DMATA[5:0] | DMA transfer access start address<br>This field defines the start address of accessing the TIMERx_DMATB register by DMA. When the first access to the TIMERx_DMATB register is done, this bit-field specifies the address just accessed. And then the address of the second access to the TIMERx_DMATB register will be (start address + 0x4).<br>6'b000000: TIMERx_CTL0<br>6'b000001: TIMERx_CTL1<br>…<br>In a word: start address = TIMERx_CTL0 + DMATA*4 |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0xE4

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

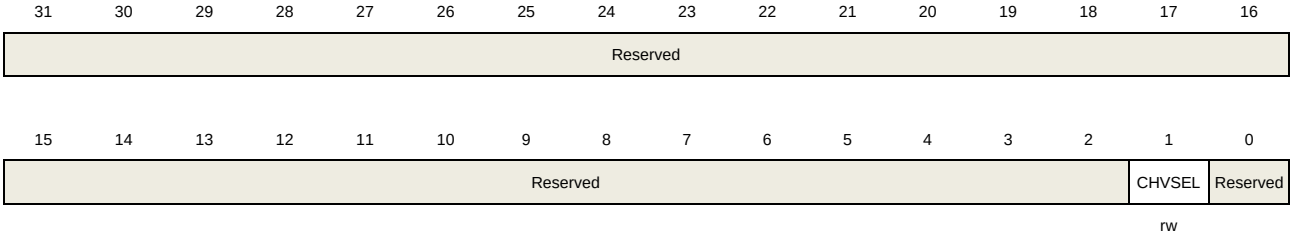
| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMATB[31:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address ranges from (start address) to (start address + transfer count * 4) will be accessed.<br>The transfer count is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | CHVSEL | Write CHxVAL register selection bit<br>This bit-field is set and reset by software.<br>1: If the value to be written to the CHxVAL register is the same as the value of CHxVAL register, the write access is ignored.<br>0: No effect. |
| 0 | Reserved | Must be kept at reset value. |

# 24.3. General level3 timer (TIMERx, x=14,40,41,42,43,44)

## 24.3.1. Overview

The general level3 timer module (TIMER14/40/41/42/43/44) is a three-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level3 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level3 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which is suitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

## 24.3.2. Characteristics

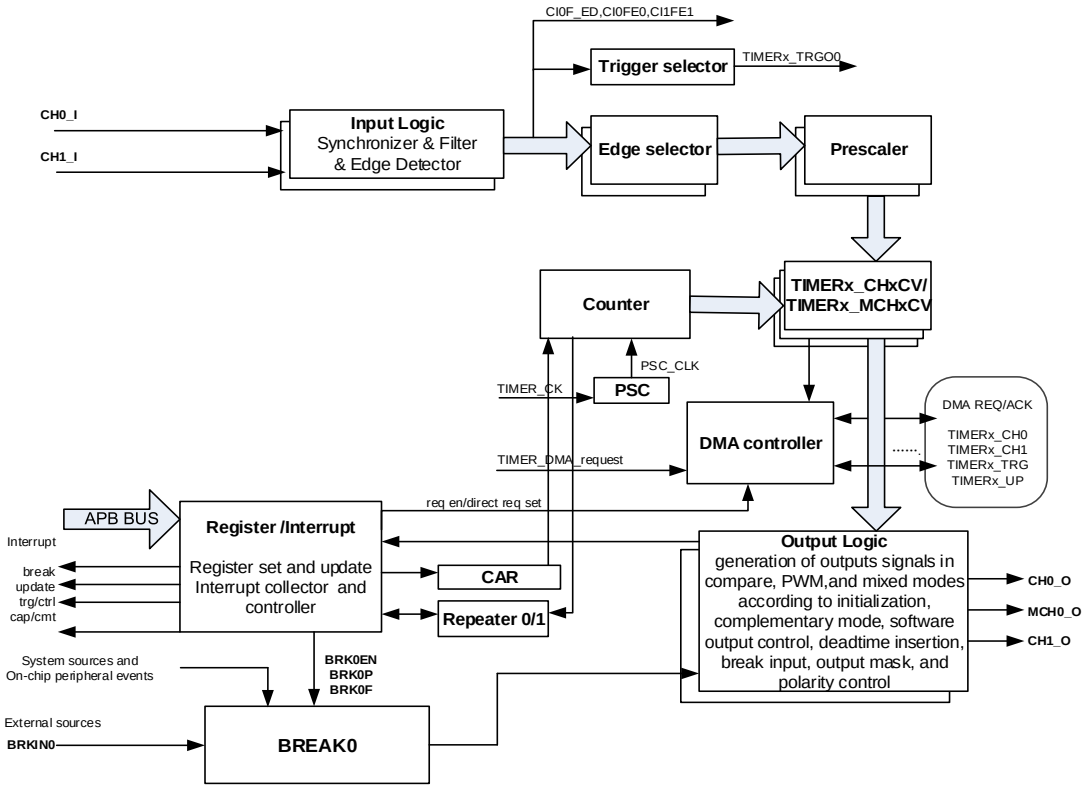
- Total channel num: 3.
- Counter width: 16-bit.
- Source of counter clock is selectable: internal clock, internal trigger, external input.
- Counter modes: count up only.
- Programmable prescaler: 16-bit. The factor can be changed on the go.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode, single pulse mode.
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input function: BREAK0.
- Interrupt output or DMA request: update, trigger event, compare/capture event, and break input.
- Daisy chaining of timer modules allows a single timer to initiate multiple timers.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer master-slave management.

## 24.3.3. Block diagram

*Figure 24-81. General level3 timer block diagram* provides details of the internal

configuration of the general level3 timer.

**Figure 24-81. General level3 timer block diagram**




## 24.3.4. Function overview

### Clock selection

The general level3 timer has the capability of being clocked by either the TIMER_CK or an alternate clock source controlled by TSCFGy[4:0] (y=3..7,15) in SYSCFG_TIMERxCFG(x=14,40,41,42,43,44) registers.

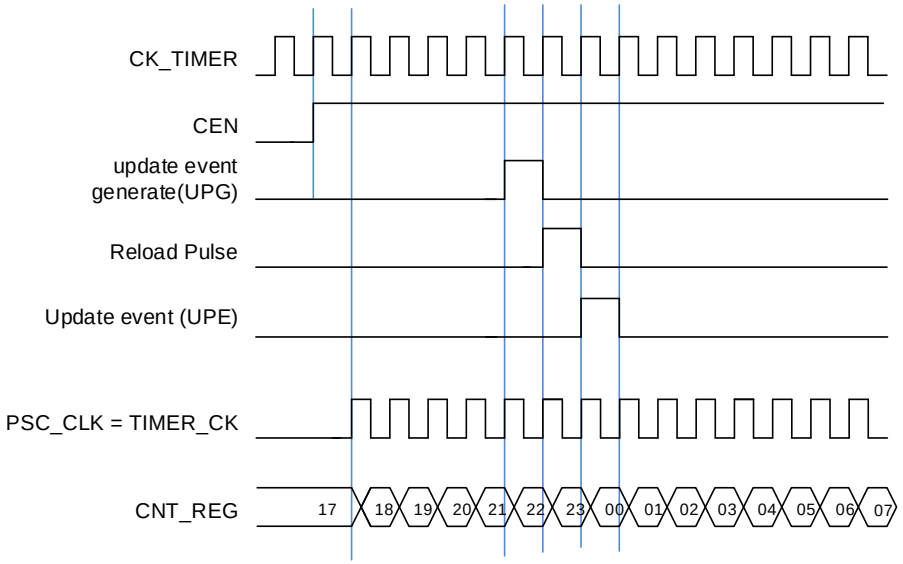
- TSCFGy[4:0] (y=3..7,15) = 5'b00000 in SYSCFG_TIMERxCFG(x=14,40,41,42,43,44) registers. Internal clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when TSCFGy[4:0] (y=3..7,15) = 5'b00000 in SYSCFG_TIMERxCFG(x=14,40,41,42,43,44) registers. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, which drives counter's prescaler to count, is equal to CK_TIMER which is from RCU.

If TSCFG6[4:0] bit-filed in SYSCFG_TIMERxCFG(x=0,7) registers are setting to a nonzero value, the prescaler is clocked by other clock sources selected in the TSCFG6[4:0] bit-filed, more details will be introduced later. When the TSCFGy[4:0] (y=3,4,5,7) are setting to an available value, the internal clock TIMER_CK is the counter prescaler driving clock source.

**Figure 24-82. Normal mode, internal clock divided by 1**




- TSCFG6[4:0] are setting to a nonzero value (external clock mode 0). External input pin is selected as timer clock source

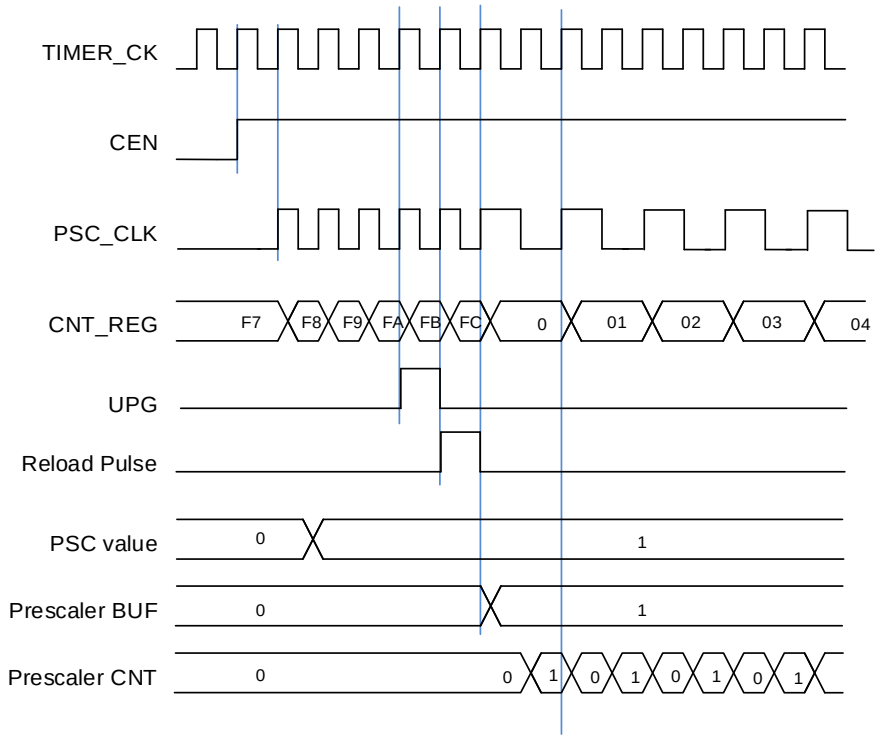
The TIMER_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CH0/TIMERx_CH1. This mode can be selected by setting TSCFG6[4:0] to 0x5~0x7.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting TSCFG6[4:0] to 0x1~0x4.

### Clock prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMERx_PSC) which can be changed on the go but is taken into account at the next update event.

**Figure 24-83. Counter timing diagram with prescaler division change from 1 to 2**




### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after (TIMERx_CREP0/1+1) times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

***Figure 24-84. Timing diagram of up counting mode, PSC=0/2*** and ***Figure 24-85. Timing diagram of up counting mode, change TIMERx_CAR on the go*** show some examples of

the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 24-84. Timing diagram of up counting mode, PSC=0/2**

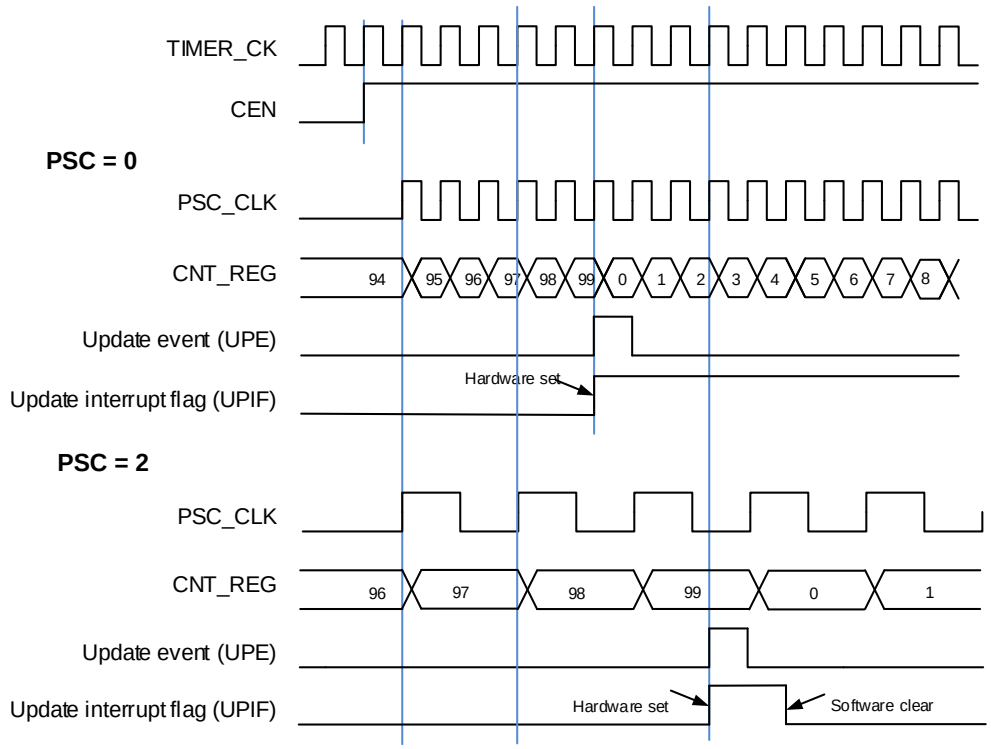
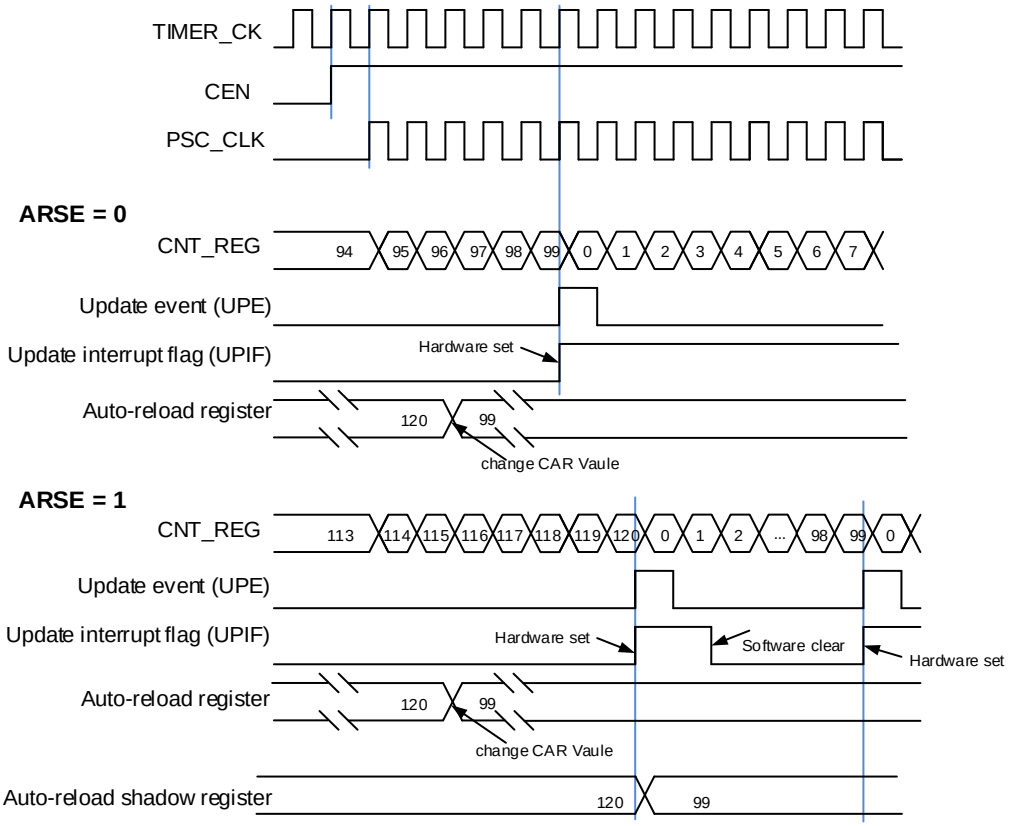



**Figure 24-85. Timing diagram of up counting mode, change TIMERx_CAR on the go**



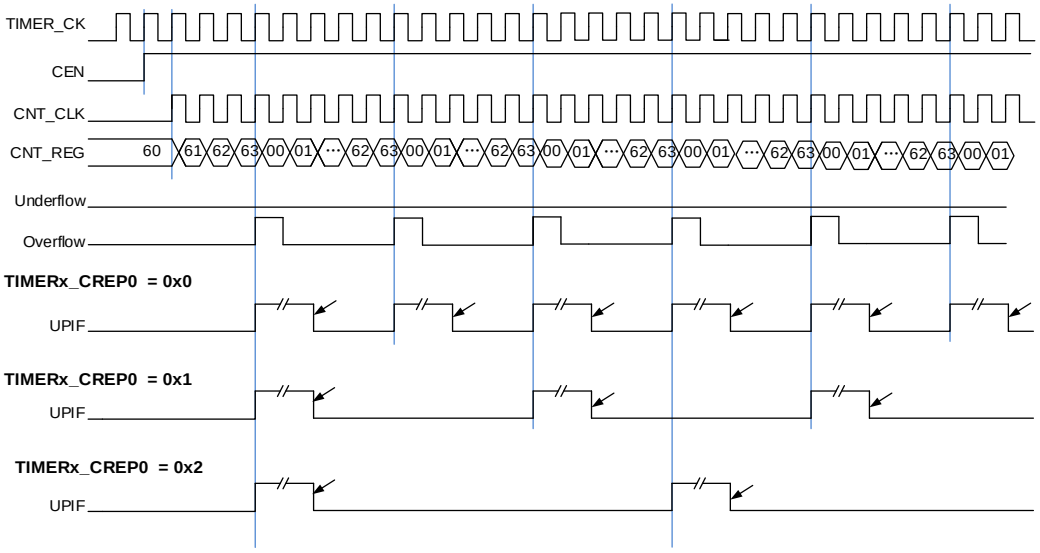
### Counter repetition

The general timer has two repetitions counter TIMERx_CREP0/1, which can be selected by configuring the CPERSEL bit in the TIMERx_CFG register. The CPEP[7:0] bit-field is 8bits, the CPEP[31:0] bit-field is 32bits and can be read on the fly.

Counter repetition is used to generator update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP0/1 in TIMERx_CREP0/1 register. The repetition counter is decremented at each counter overflow in up-counting mode.

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP0/1 in TIMERx_CREP0/1 register and generator an update event.

**Figure 24-86. Repetition timechart for up-counter**




### Capture/compare channels

The general level3 timer has three independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

When the channels are used for input, channel 0 and multi mode channel 0 can perform input capture independently; when the channels are used for comparison output, the channel 0 and multi mode channel 0 can output independent and complementary outputs.

- **Input capture mode**

When MCHxMSEL=2'b00(independent mode), channel x and multi mode channel x can perform input capture independently.

Input capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV/ TIMERx_MCHxCV(x=0, 1) registers, at the same time the CHxIF/ MCHxIF(x=0, 1) bits are

set and the channel interrupt is generated if it is enabled when CHxIE/ MCHxIE =1(x=0, 1).

**Figure 24-87. Input capture logic for channel 0**

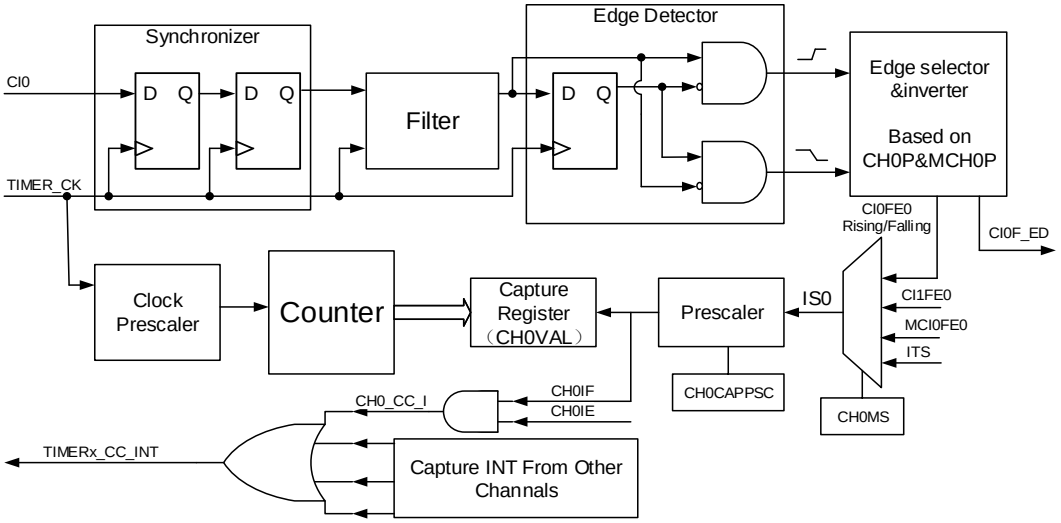
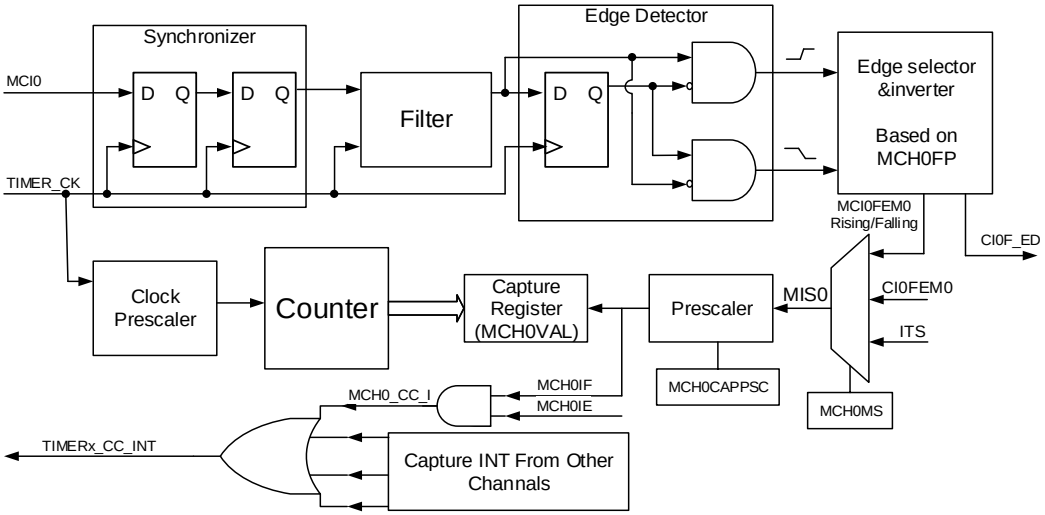



**Figure 24-88. Input capture logic for multi mode channel 0**




The input signals of channelx (CIx/ MCIx) are the TIMERx_CHx/ TIMERx_MCHxCV signal.

First, the input signal of channel (CIx/ MCIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP/ MCHxP or MCHxFP bits. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS/ MCHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV/ TIMERx_MCHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT bit in TIMERx_CHCTL0 register and MCHxCAPFLT bit in TIMERx_MCHCTL0 register).

Based on the input signal and quality of requested signal, configure compatible

CHxCAPFLT or MCHxCAPFLT bit.

**Step2**: Edge selection (CHxP and MCHxP bits in TIMERx_CHCTL2 register, MCHxFP[1:0] bits in TIMERx_MCHCTL2 register).

Rising edge or falling edge, choose one by configuring CHxP and MCHxP bits or MCHxFP[1:0] bits.

**Step3**: Capture source selection (CHxMS bit in TIMERx_CHCTL0 register, MCHxMS bit in TIMERx_MCHCTL0 register).

As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x000 or MCHxMS != 0x000) and TIMERx_CHxCV/ TIMERx_MCHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN bits, MCHxIE and MCHxDEN bits in TIMERx_DMAINTEN).

Enable the related interrupt to get the interrupt and DMA request.

**Step5**: Capture enable (CHxEN and MCHxEN bits in TIMERx_CHCTL2).

**Result**: When the wanted input signal is captured, TIMERx_CHxCV/ TIMERx_MCHxCV will be set by counter's value and CHxIF/ MCHxIF bit is asserted. If the CHxIF/ MCHxIF bit is 1, the CHxOF/ MCHxOF bit will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN bits, MCHxIE and MCHxDEN bits in TIMERx_DMAINTEN.

**Direct generation**: A DMA request or interrupt is generated by setting CHxG directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx and TIMERx_MCHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 3'b001 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 3'b010 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty cycle.

## Output compare mode

*Figure 24-89. Output compare logic (when MCHxMSEL = 2'00, x=0)*, *Figure 24-90. Output compare logic (when MCHxMSEL = 2'11, x=0)* and *Figure 24-91. Output compare logic (x=1)* show the logic circuit of output compare mode.

**Figure 24-89. Output compare logic (when MCHxMSEL = 2'00, x=0)**

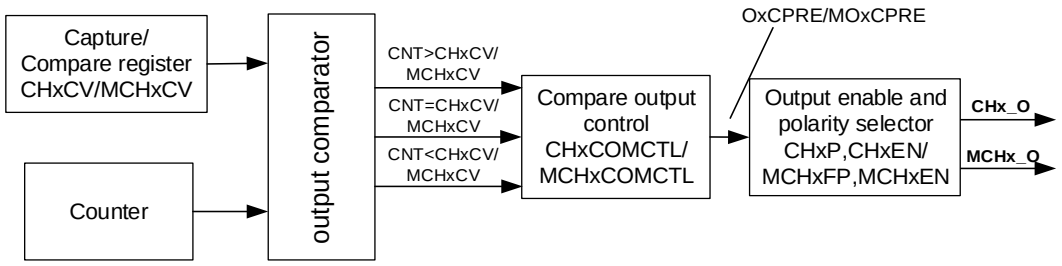


**Figure 24-90. Output compare logic (when MCHxMSEL = 2'11, x=0)**

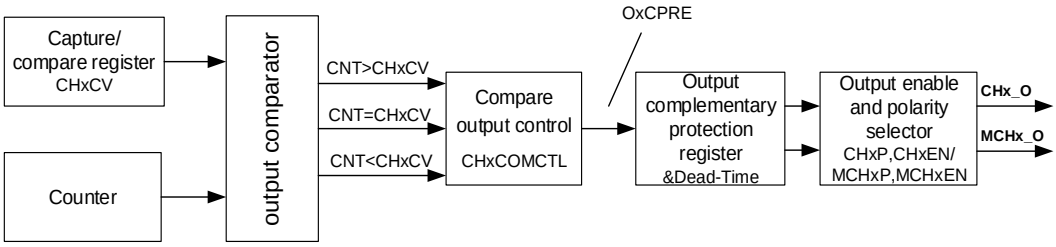
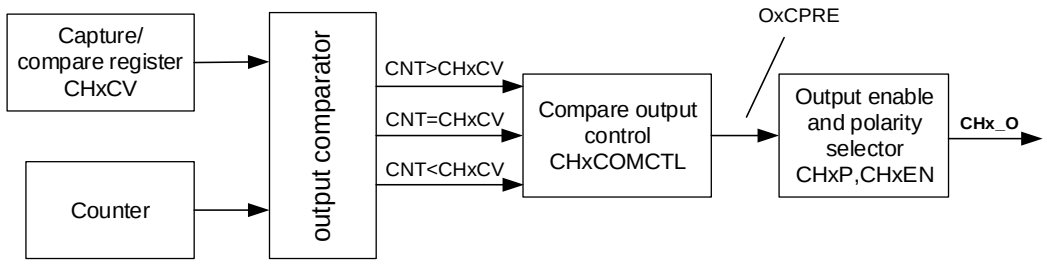



**Figure 24-91. Output compare logic (x=1)**




The relationship between the channel output signal CHx_O/MCHx_O and the OxCPRE/ MOxCPRE signal (more details refer to ***Channel output prepare signal***) is described as blew(the active level of OxCPRE is high and the active level of MOxCPRE is high).

- When MCHxMSEL=2'b00 (in TIMERx_CTL2 register), the MCHx_O output is independent from the CHx_O output. The output level of CHx_O depends on OxCPRE signal, CHxP bit and CHxEN bit (please refer to the TIMERx_CHCTL2 register for more details). The output level of MCHx_O depends on MOxCPRE signal, MCHxFP[1:0] bits and MCHxEN bit (please refer to the TIMERx_MCHCTL2 and TIMERx_CHCTL2 registers for more details). Please refer to ***Figure 24-89. Output compare logic (when MCHxMSEL = 2'00, x=0)***.
- When MCHxMSEL=2'b11, the MCHx_O output is the inverse of the CHx_O output. The output level of CHx_O/MCHx_O depends on OxCPRE signal, CHxP/ MCHxP bits and CHxEN/MCHxEN bits. Please refer to ***Figure 24-90. Output compare logic (when MCHxMSEL = 2'11, x=0)***.

For examples (the MCHx_O output is independent from the CHx_O output):

3) Configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled):
   If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
   If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

4) Configure MCHxP=1 (the active level of MCHx_O is low, contrary to MOxCPRE), MCHxEN=1 (the output of MCHx_O is enabled):
   If the output of MOxCPRE is active(high) level, the output of MCHx_O is active(low) level;

If the output of MOxCPRE is inactive(low) level, the output of MCHx_O is active(high) level.

When MCHxMSEL=2'b11 and CHx_O and MCHx_O are output at the same time, the specific outputs of CHx_O and MCHx_O are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register. Please refer to ***Outputs complementary*** for more details.

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV/ TIMERx_MCHxCV register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL/ MCHxCOMCTL. When the counter reaches the value in the TIMERx_CHxCV/ TIMERx_MCHxCV register, the CHxIF/ MCHxIF bit will be set and the channel (n) interrupt is generated if CHxIE/ MCHxIE = 1. And the DMA request will be asserted, if CHxDEN/ MCHxDEN =1.

So, the process can be divided into several steps as below:

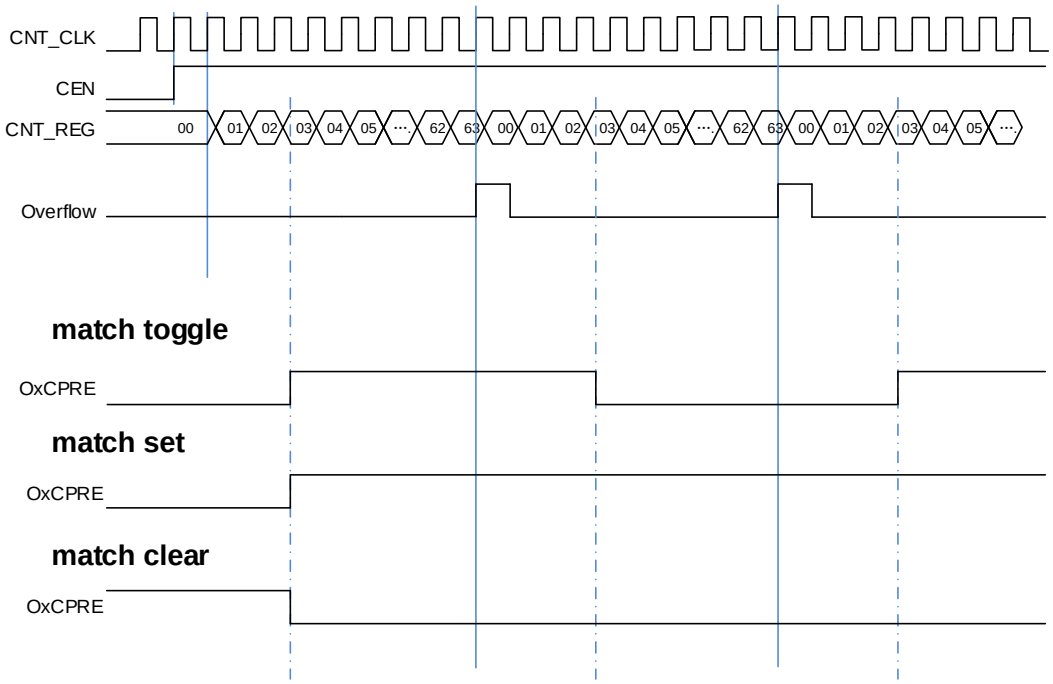
**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.
**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN/ MCHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL/ MCHxCOMCTL.
- Select the active polarity by CHxP/MCHxP/ MCHxFP.
- Enable the output by CHxEN/ MCHxEN.

**Step3:** Interrupt/DMA request enable configuration by CHxIE/ MCHxIE /CHxDEN/ MCHxDEN.
**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV/ TIMERx_MCHxCV.
The TIMERx_CHxCV/ TIMERx_MCHxCV can be changed onging to meet the expected waveform.
**Step5:** Start the counter by configuring CEN to 1.

***Figure 24-92. Output-compare in three modes*** shows the three compare modes: toggle/set/clear. CARL=0x63, CHxVAL=0x3.

**Figure 24-92. Output-compare in three modes**



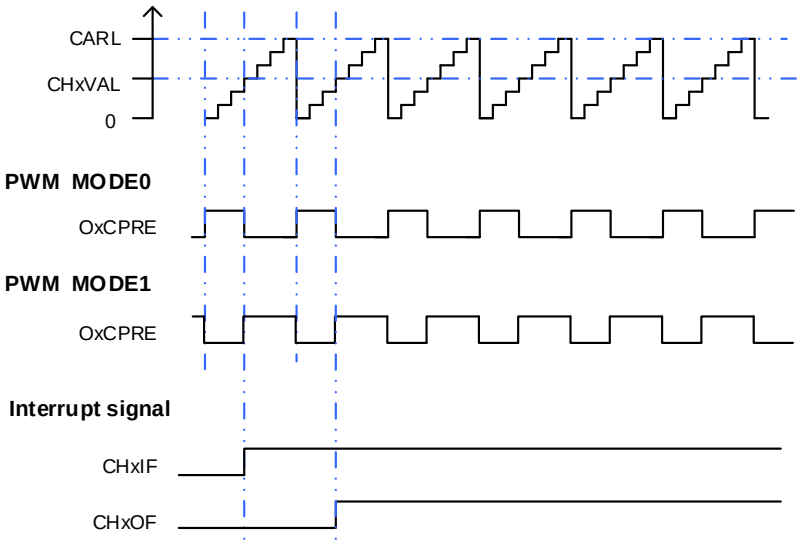

## PWM mode

In the PWM output mode (by setting the CHxCOMCTL/ MCHxCOMCTL bit to 4'b0110 (PWM mode 0) or to 4'b0111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV/ TIMERx_MCHxCV registers.

The EAPWM's period is determined by TIMERx_CAR and the duty cycle is determined by TIMERx_CHxCV/ TIMERx_MCHxCV. ***Figure 24-93. PWM mode timechart*** shows the EAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx_CHxCV/ TIMERx_MCHxCV is greater than the value of TIMERx_CAR, the output will be always active in PWM mode 0 (CHxCOMCTL/ MCHxCOMCTL =4'b0110). And if the value of TIMERx_CHxCV/ TIMERx_MCHxCV is greater than the value of TIMERx_CAR, the output will be always inactive in PWM mode 1 (CHxCOMCTL/ MCHxCOMCTL =4'b0111).

**Figure 24-93. PWM mode timechart**




## Composite PWM mode

In the Composite PWM mode (CHxCPWMEN = 1'b1, CHxMS[2:0] = 3'b000 and CHxCOMCTL = 4'b0110 or 4'b0111), the PWM signal output in channel x (x=0, 1) is composited by CHxVAL and CHxCOMVAL_ADD bits.

If CHxCOMCTL = 4'b0110 (PWM mode 0) and DIR = 1'b0 (up counting mode), the channel x output is forced low when the counter matches the value of CHxVAL. It is forced high when the counter matches the value of CHxCOMVAL_ADD.

If CHxCOMCTL = 4'b0111 (PWM mode 1) and DIR = 1'b0 (up counting mode), the channel x output is forced high when the counter matches the value of CHxVAL. It is forced low when the counter matches the value of CHxCOMVAL_ADD.

The PWM period is determined by (CARL + 0x0001) and the PWM pulse width is determined by the following table.

**Table 24-14.The Composite PWM pulse width**

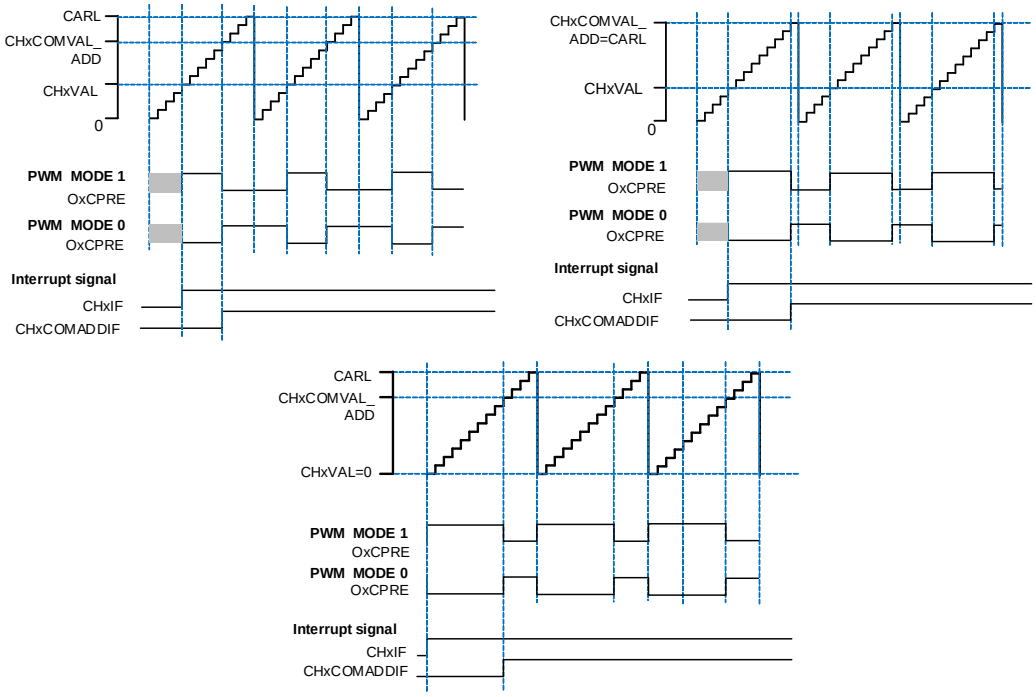
| Condition | Mode | PWM pulse width |
|---|---|---|
| CHxVAL < CHxCOMVAL_ADD ≤ CARL | PWM mode 0 | (CARL + 0x0001) + (CHxVAL – CHxCOMVAL_ADD) |
| | PWM mode 1 | (CHxCOMVAL_ADD – CHxVAL) |
| CHxCOMVAL_ADD < CHxVAL ≤ CARL | PWM mode 0 | (CHxVAL - CHxCOMVAL_ADD) |
| | PWM mode 1 | (CARL + 0x0001) + (CHxCOMVAL_ADD – CHxVAL) |
| (CHxVAL = CHxCOMVAL_ADD ≤ CARL) or (CHxVAL > CARL > CHxCOMVAL_ADD) | PWM mode 0 (up counting) or PWM mode 1 (down counting) | 100% |
| | PWM mode 0 (down counting) or PWM mode 1 (up counting) | 0% |
| CHxCOMVAL_ADD > CARL > CHxVAL | PWM mode 0(up counting) or PWM mode 1(down counting) | 0% |
| | PWM mode 0(down counting) or PWM mode 1(up counting) | 100% |
| (CHxVAL>CARL) and (CHxCOMVAL_ADD > CARL) | - | The output of CHx_O is keeping |

When the counter reaches the value of CHxVAL, the CHxIF bit is set and the channel x interrupt is generated if CHxIE = 1, and the DMA request will be asserted, if CHxDEN=1. When the counter reaches the value of CHxCOMVAL_ADD, the CHxCOMADDIF bit is set (this flag just used in composite PWM mode, when CHxCPWMEN=1) and the channel x additional compare interrupt is generated if CHxCOMADDIE = 1 (Only interrupt is generated, no DMA request is generated).

According to the relationship among CHxVAL, CHxCOMVAL_ADD and CARL, it can be divided into four situations:

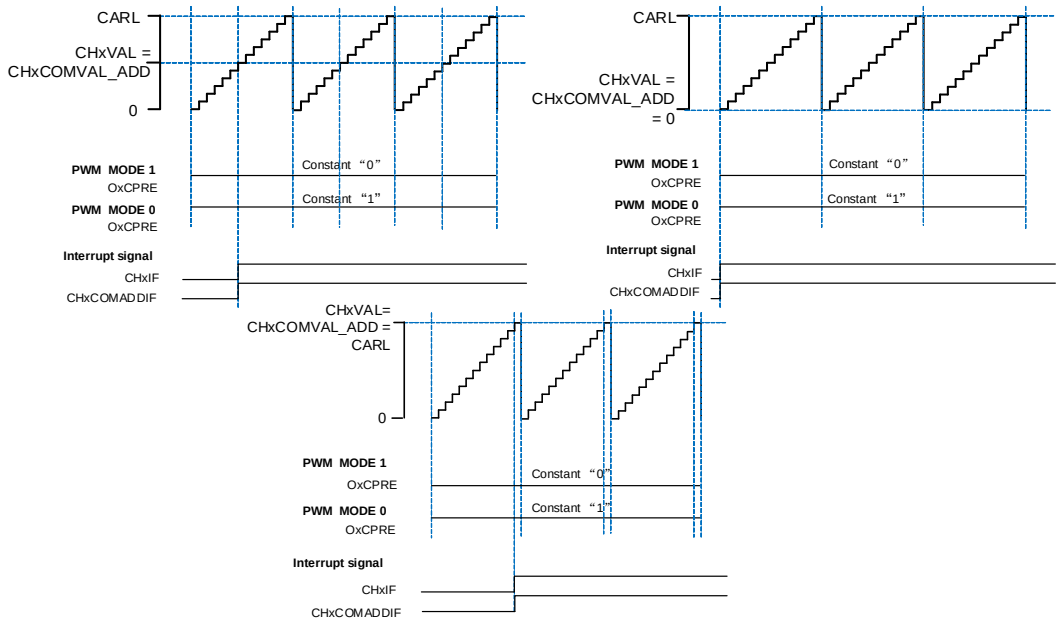
1) CHxVAL < CHxCOMVAL_ADD, and the values of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-94. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD)**



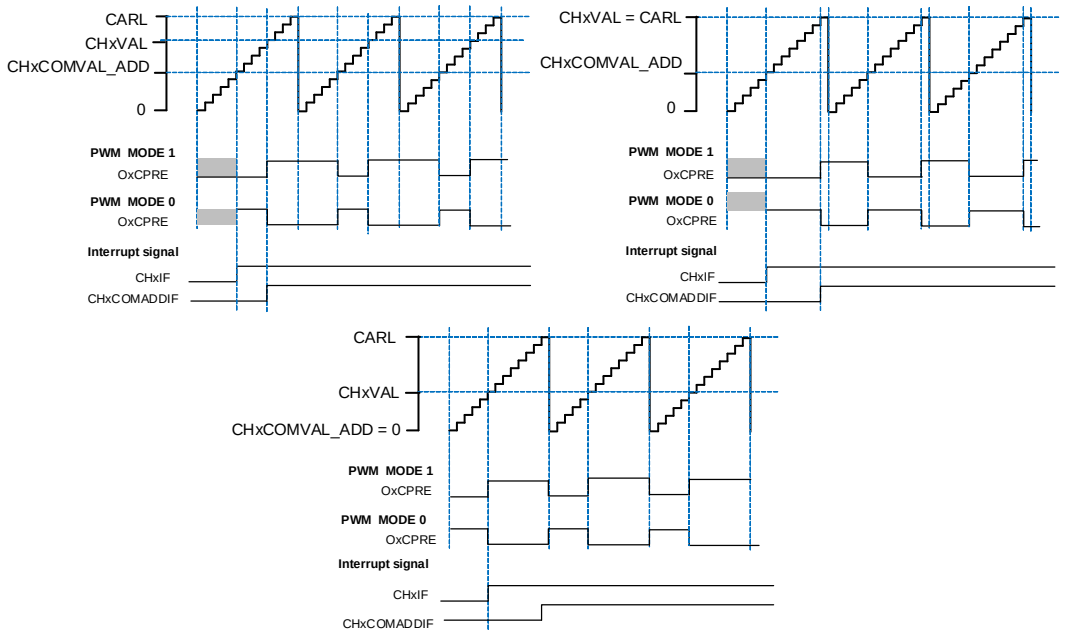

2) CHxVAL = CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-95. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD)**



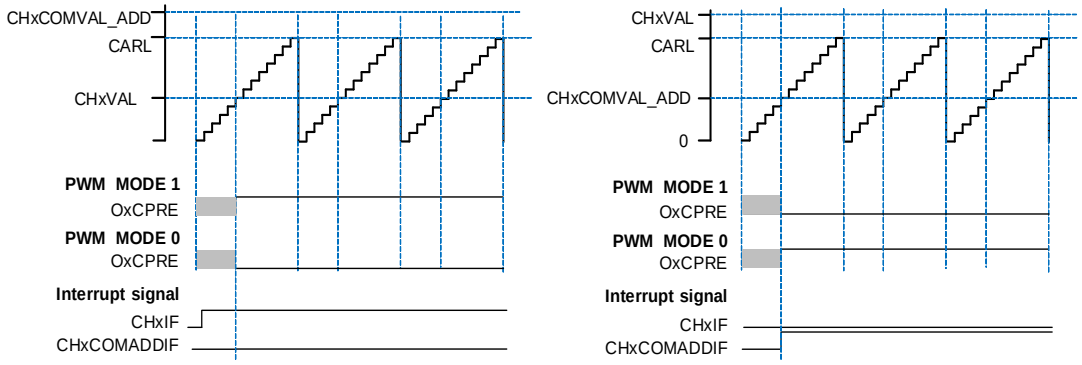

3) CHxVAL > CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

**Figure 24-96. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD)**




4) One of the value of CHxVAL and CHxCOMVAL_ADD exceeds CARL.

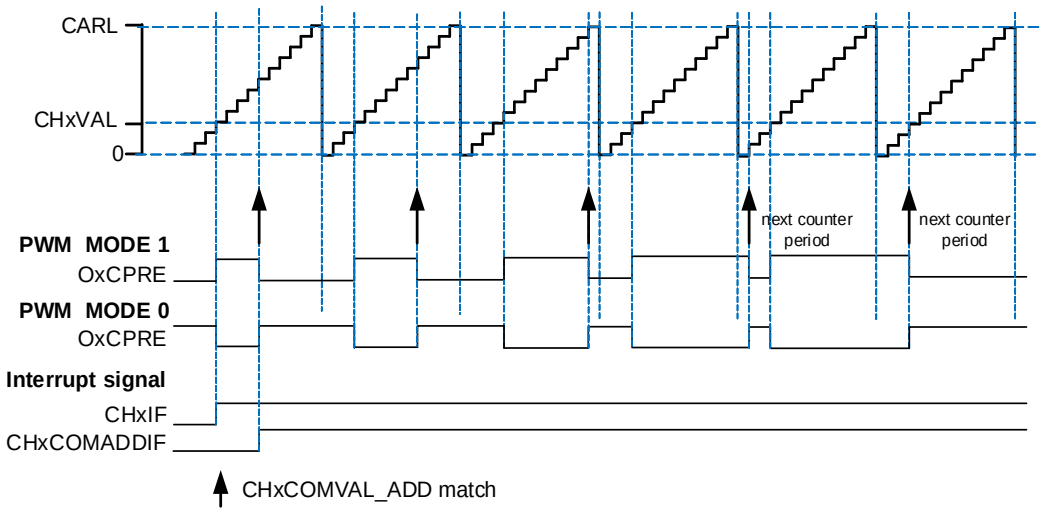
**Figure 24-97. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL**




The composite PWM mode is intended to support the generation of PWM signals where the period is not modified while the signal is being generated, but the duty cycle will be varied. ***Figure 24-98. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD*** shows the PWM output and interrupts waveform.

In some cases, the CHxCOMVAL_ADD match can happen on the next counter period (the value of CHxCOMVAL_ADD was written after the counter reaches the value of CHxVAL, and the value of CHxCOMVAL_ADD was less than or equal to the CHxVAL).

**Figure 24-98. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD**




If more than one channels are configured in composite PWM mode, it is possible to fix an offset for the channel x match edge of each pair with respect to other channels. This behavior is useful in the generation of lighting PWM control signals where it is desirable that edges are not coincident with each other pair to help eliminate noise generation. The CHxVAL register value is the shift of the PWM pulse with respect to the beginning of counter period.

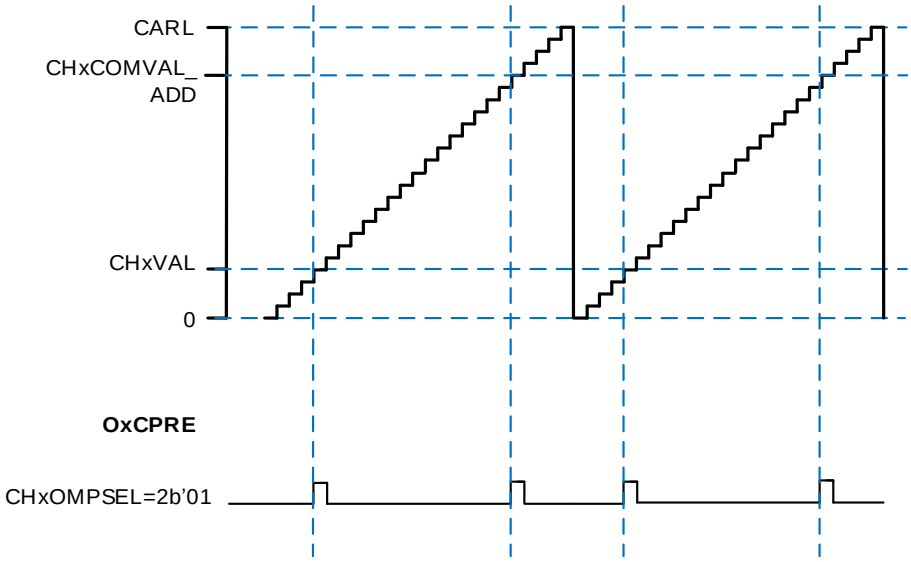
### Output match pulse select

Basing on that CHx_O (x=0, 1) outputs are configured by CHxCOMCTL[3:0](x=0, 1) bits when the match events occur, the output signal is configured by CHxOMPSEL[1:0](x=0, 1) bits to be normal or a pulse.

When the match events occur, the CHxOMPSEL[1:0](x=0, 1) bits are used to select the output of OxCPRE which drives CHx_O:

- CHxOMPSEL = 2'b00, the OxCPRE signal is output normally with the configuration of CHxCOMCTL[3:0] bits;
- CHxOMPSEL = 2'b01, only the counter is counting up, the OxCPRE signal is output a pulse when the match events occur, and the pulse width is one CK_TIMER clock cycle.

**Figure 24-99. CHx_O output with a pulse in edge-aligned mode (CHxOMPSEL =2'b00)**




### Channel output prepare signal

As is shown in ***Figure 24-89. Output compare logic (when MCHxMSEL = 2'00, x=0)***, ***Figure 24-90. Output compare logic (when MCHxMSEL = 2'11, x=0)*** and ***Figure 24-91. Output compare logic (x=1)***, when TIMERx is configured in compare match output mode, a middle signal named OxCPRE or MOxCPRE (channel x output or multi mode channel x output prepare signal) will be generated before the channel outputs signal.

The OxCPRE and MOxCPRE signal have several types of output function. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit and the MOxCPRE signal type is defined by configuring the MCHxCOMCTL bit.

Take OxCPRE as an example for description below, these include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/ PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/ 0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Outputs complementary

The outputs of CHx_O and MCHx_O have two situations:

- MCHxMSEL=2'b00: The MCHx_O output is independent from the CHx_O output;
- MCHxMSEL=2'b11: The outputs of MCHx_O and CHx_O are complementary and the MCHxCOMCTL bits are not used in the generation of the MCHx_O output.

Function of complementary is for a pair of channels, CHx_O and MCHx_O, the two output signals cannot be active at the same time. TIMERx's one pair of channel has this function. The complementary signals CHx_O and MCHx_O are controlled by a group of parameters: the CHxEN and MCHxEN bits in the TIMERx_CHCTL2 register, the POEN, ROS and IOS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register. The output polarity is determined by CHxP and MCHxP bits in the TIMERx_CHCTL2 register.

When the the outputs of CHx_O and MCHx_O are complementary, there are three situations: output enable、output off-state and output disabled. The details are shown in ***Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)***.

**Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)**

| Complementary Parameters | | | | | Output Status | |
|---|---|---|---|---|---|---|
| POEN | ROS | IOS | CHxEN | MCHxEN | CHx_O | MCHx_O |
| 0 | 0/1 | 0 | 0 | 0 | CHx_O / CHx_ON = LOW<br>CHx_O / CHx_ON output disable(1). | |
| | | | | 1 | CHx_O/ CHx_ON output "off-state" (2):<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3) | |
| | | | 1 | 0 | | |
| | | | | 1 | | |
| | | 1 | x | x | CHx_O/ CHx_ON output "off-state":<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. | |
| 1 | 0 | 0/1 | 0 | 0 | CHx_O/MCHx_O = LOW<br>CHx_O/MCHx_O output disable. | |
| | | | | 1 | CHx_O = LOW<br>CHx_O output disable. | MCHx_O=OxCPRE⊕(4)MCHxP<br>MCHx_O output enable. |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | MCHx_O = LOW<br>MCHx_O output disable. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | MCHx_O = (! OxCPRE)(5)⊕MCHxP.<br>MCHx_O output enable. |
| | 1 | | 0 | 0 | CHx_O = CHxP<br>CHx_O output "off-state". | MCHx_O = MCHxP<br>MCHx_O output "off-state". |
| | | | | 1 | CHx_O = CHxP<br>CHx_O output "off-state" | MCHx_O=OxCPRE⊕MCHxP<br>MCHx_O output enable |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | MCHx_O = MCHxP<br>MCHx_O output "off-state". |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | MCHx_O = (! OxCPRE)⊕MCHxP<br>MCHx_O output enable. |

**Note:**

(1) output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) "off-state": CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (! OxCPRE)：the complementary output of the OxCPRE signal.

### Dead time insertion

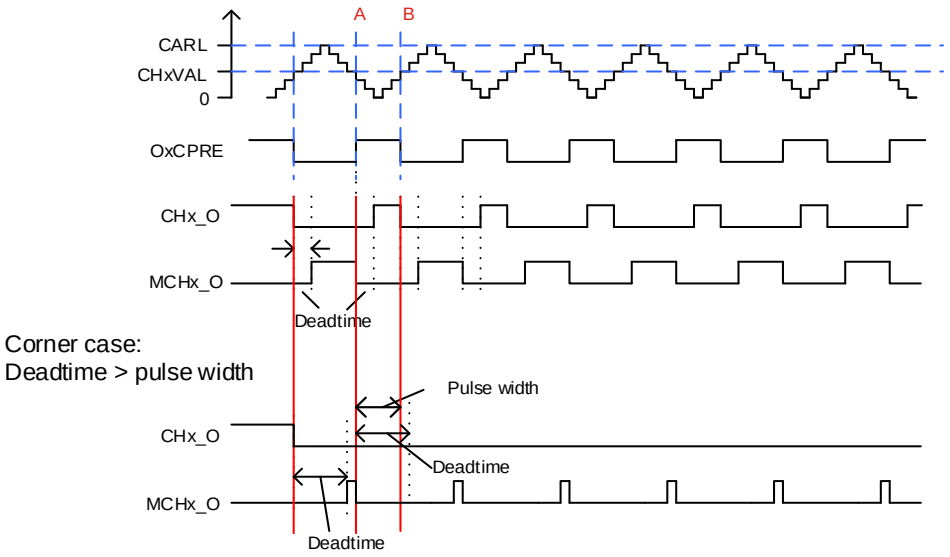
The dead time insertion is enabled when MCHxMSEL=2'b11 and both CHxEN and MCHxEN are configured to 1'b1, it is also necessary to configure POEN to 1. The field named DTCFG defines the dead time delay that can be used for all channels. Refer to the ***Complementary channel protection register (TIMERx_CCHP)*** for details about the delay time.

The dead time delay insertion ensures that two complementary signals are not active at the same time.

When the channel x match event (TIMERx_CNT = CHxVAL) occurs, OxCPRE will be toggled in PWM mode 0. At point A in ***Figure 24-100. Complementary output with dead-time insertion***, CHx_O signal remains at the low level until the end of the dead time delay, while MCHx_O signal will be cleared at once. Similarly, at point B when the channelx match event (TIMERx_CNT = CHxVAL) occurs again, OxCPRE is cleared, and CHx_O signal will be cleared at once, while MCHx_O signal remains at the low level until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example: the dead time delay is greater than or equal to the duty cycle of the CHx_O signal, then the CHx_O signal is always inactive (As shown in ***Figure 24-100. Complementary output with dead-time insertion***).

**Figure 24-100. Complementary output with dead-time insertion**




### Break function

The MCHx_O output is the inverse of the CHx_O output when the MCHxMSEL=2'b11 (and the MCHxCOMCTL bits are not used in the generation of the MCHx_O output). In this case, CHx_O and MCHx_O signals cannot be set to active level at the same time.

The general level3 timers have the BREAK0 function. The BREAK0 function can be enabled by setting the BRK0EN bit in the TIMERx_CCHP register. The break input polarity is

configured by the BRK0P bit in TIMERx_CCHP register, the input is active on level.

In BREAK0 function, CHx_O and MCHx_O are controlled by the POEN, OAEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register.

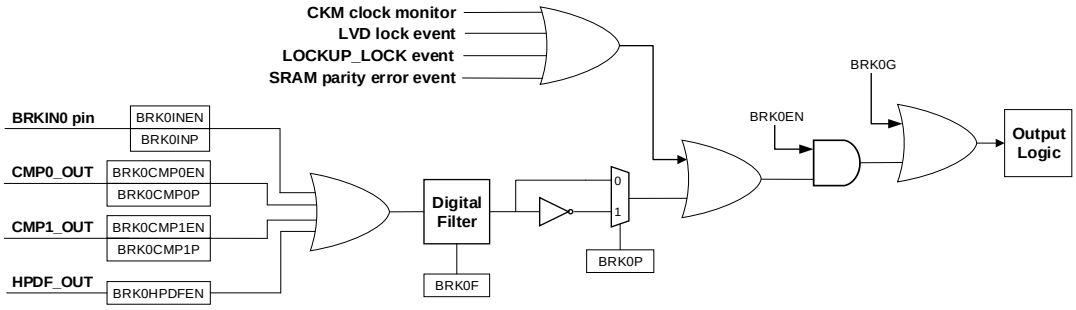
The break event is the result of logic ORed of all sources. The BREAK0 function can handle three types of event sources:

- External sources: coming from BRKIN0 input;
- System sources: HXTAL stuck event which is generated by Clock Monitor CKM in RCU, LVD lock event, Cortex®-M7 LOCKUP_LOCK event or SRAM parity error event;
- On-chip peripheral events: input by comparator output or HPDF watchdog output.

BREAK0 events can also be generated by software using BRK0G bit in the TIMERx_SWEVG register.

Refer to ***Figure 24-101. BREAK0 function logic diagram***, BRKIN0 can select GPIO pins from the TRIGSEL module, which can select by ***Trigger selection for TIMER14_BRKIN register (TRIGSEL_TIMER14BRKIN)***.

**Figure 24-101. BREAK0 function logic diagram**

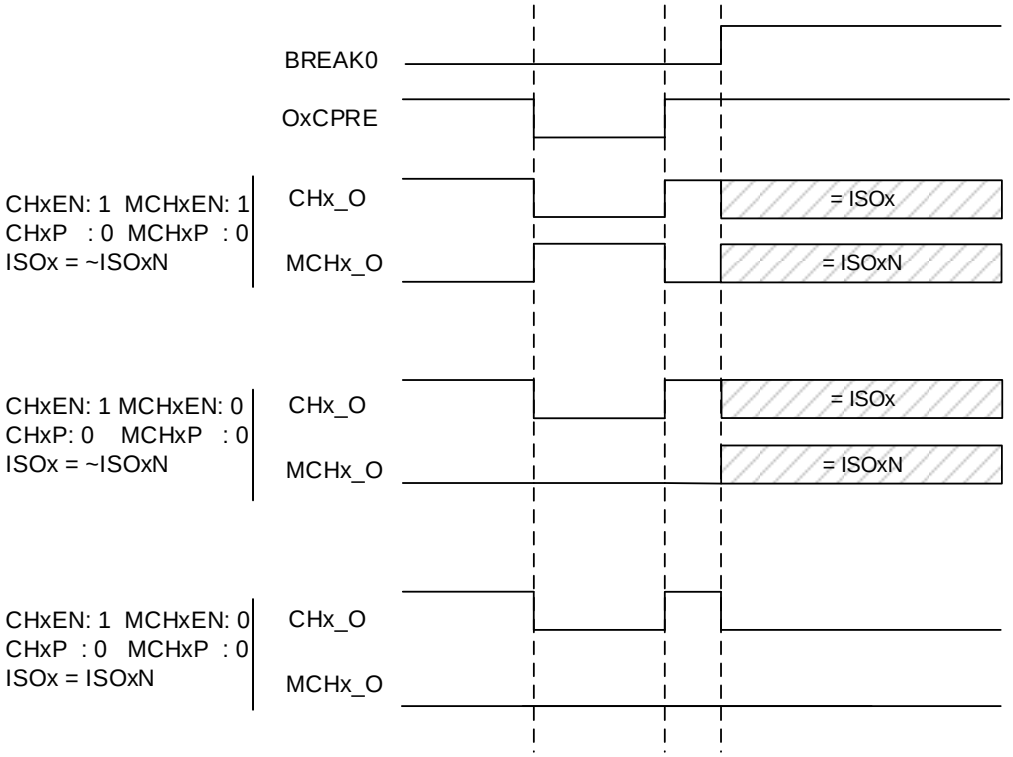



When a BREAK0 event occurs, the outputs are force at an inactive level, or at a predefined level (either active or inactive) after a deadtime duration.

When the MCHxMSEL = 2'b11 and a break occurs, the POEN bit is cleared asynchronously. As soon as POEN is 0, the level of the CHx_O and MCHx_O outputs are determined by the ISOx and ISOxN bits in the TIMERx_CTL1 register. If IOS = 0, the timer releases the enable output, otherwise, the enable output remains high. When IOS=1, the output behavior of the channel is shown in ***Figure 24-102. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1)***. The complementary outputs are first in the reset state, and then the dead time generator is reactivated to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead time.

**Figure 24-102. Output behavior of the channel in response to BREAK0 (the break input**

**high active and IOS=1)**




When a break occurs, the BRKIF bit in the TIMERx_INTF register will be set. If BRKIE is 1, an interrupt will be generated.

### Locked break function

The BRKIN0 input pin of general timer have the locked break function, this function can be enabled by setting the BRK0LK bit in the TIMERx_CCHP register.

When the locked break function is enabled, the BRKIN0 pins need to be configured to open-drain output mode with low level active (BRK0P=0 and BRK0IN0P=0). When any break source requests occur, the BRKIN0 pin can be forced to low level. If the break input polarity is active high (BRK0P=1 and BRK0IN0P =1), the locked break function is invalid.

When the break function is enabled (the BRK0EN =1), the BRKIN0 pin can be forced to low level with the BRK0G bit setting to 1 by software.

When the break function is disabled (the BRK0EN =0), setting the BRK0G bit will have no effect on the BRKIN0 pin. The BRK0F bit will set and the channel outputs will be in a safe state.

The BRKIN0 pin can be released by setting the BRK0REL bit in the TIMERx_CCHP register. When the break input sources are inactive, the BRK0REL bit will cleared by hardware and the BRKIN0 pin will restore the locked break function.

In the following two cases, the BRKIN0 pin cannot be released:

- Break input sources are active: the BRK0REL bit is set to 1 and the BRKIN0 pin locked

break function is released. The break events are still active, because the break input sources are still active.

- POEN=1: when the channel outputs are enabled, the BRKIN0 pin cannot be released even if the BRK0REL is set.

**Table 24-16. Break function input pins locked/ released conditions**

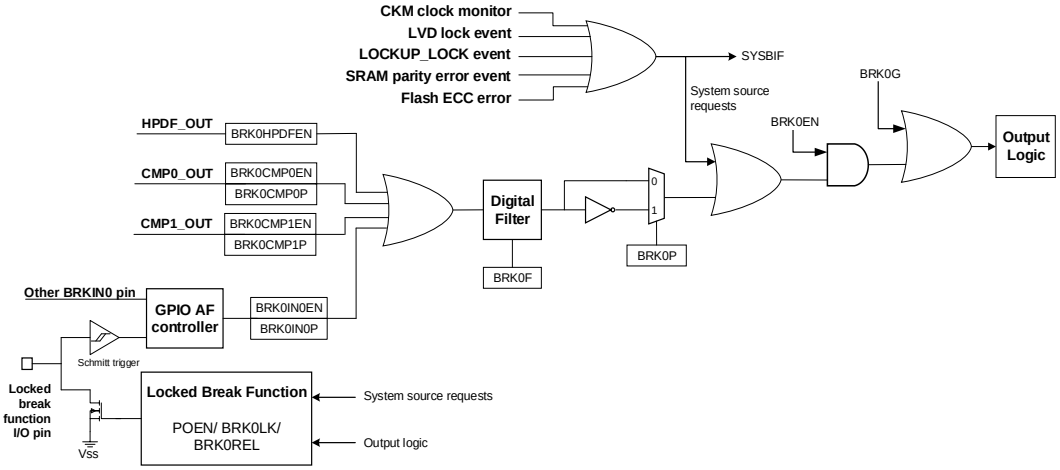
| POEN | BRK0LK | BRK0REL | Break input pin state |
|---|---|---|---|
| 0 | 1 | 0 | Locked |
| | 1 | 1 | Released |

The locked break function of the BREAK0 input pin BRKIN0 is enabled by default (BRK0REL=0). When the BREAK0 events occur, the following steps can be used to reconfigure the locked break function:

- Set the BRK0REL bit to 1 and released the BRKIN0 pin;
- The software waits for the system break sources inactive, and then clears the SYSBIF flag;
- The software polls the state of BRK0REL bit, until the BRK0REL bit is cleared (cleared by hardware).

Then the locked break function of BREAK0 input pin is re-enabled, and the channel outputs can be restored by setting the POEN bit to 1 by software.

**Figure 24-103. BRKIN0 pin logic with BREAK0 function**

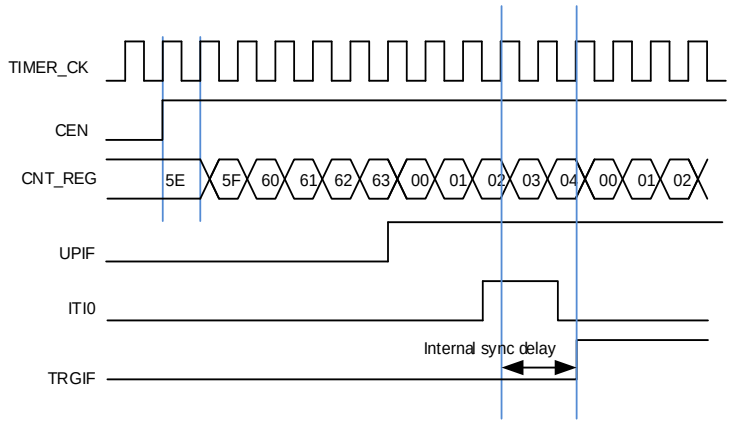
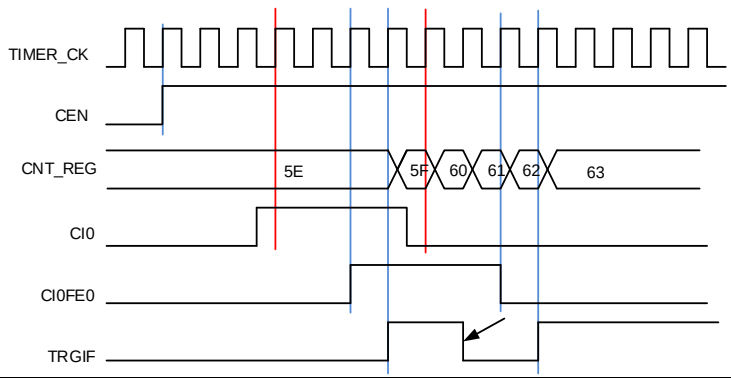



## Master-slave management

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode and so on, which is selected by the TSCFGy[4:0] (y=3..7) in SYSCFG_TIMERxCFG(x=14,40,41,42,43,44).

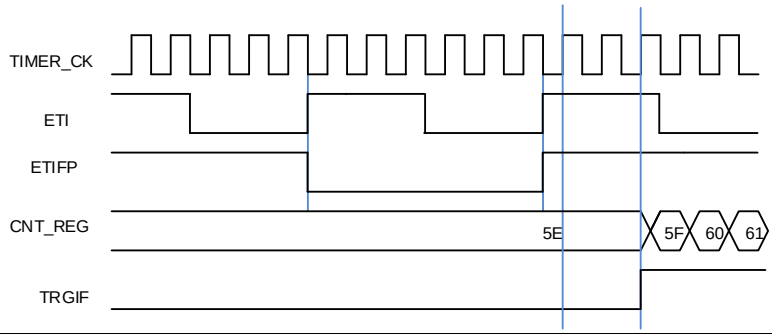
**Table 24-17. Slave mode example table**

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| **LIST** | TSCFGy[4:0]<br>y=3: restart mode<br>y=4: pause mode | TSCFGy[4:0]<br>00000:Mode disable<br>00001: ITI0 | If you choose the CIxFEx(x=0, 1) or MCIxFEMx(x=0), | For the ITIx no filter and prescaler can be used.<br>For the CIx/ MCIx, configure |

<table>
<tr><th></th><th>Mode Selection</th><th>Source Selection</th><th>Polarity Selection</th><th>Filter and Prescaler</th></tr>
<tr><td></td><td>y=5: event mode<br>y=6: external clock mode 0<br>y=7: restart + event mode</td><td>00010: ITI1<br>00011: ITI2<br>00100: ITI3<br>00101: CI0F_ED<br>00110: CI0FE0<br>00111: CI1FE1<br>10010: MCI0FEM0<br>10011: ITI14</td><td>configure the CHxP, MCHxP and MCHxFP for the polarity selection and inversion.</td><td>Filter by CHxCAPFLT/ MCHxCAPFLT, no prescaler can be used.</td></tr>
<tr><td rowspan="2">Exam1</td><td>Restart mode<br>The counter can be clear and restart when a rising trigger input.</td><td>TSCFG3[4:0]<br>5’b00001, ITI0 is the selection.</td><td>For ITI0, no polarity selector can be used.</td><td>For the ITI0, no filter and prescaler can be used.</td></tr>
<tr><td colspan="4">Figure 24-104. Restart mode<br> 
</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode<br>The counter can be paused when the trigger input is low.</td><td>TSCFG4[4:0] =5’b00110, CI0FE0 is the selection.</td><td>[MCHxP=0, CH0P=0] no inverted. Capture will be sensitive to the rising edge only.</td><td>Filter is bypass in this example.</td></tr>
<tr><td colspan="4">Figure 24-105. Pause mode<br> 
</td></tr>
<tr><td>Exam3</td><td>Event mode<br>The counter will start to count when</td><td>TSCFG5[4:0] =5’b01000,</td><td>ETP = 0, the polarity of ETI does not change.</td><td>ETPSC = 1, ETI is divided by 2.<br>ETFC = 0, ETI does not</td></tr>
</table>

| | Mode Selection | Source Selection | Polarity Selection | Filter and Prescaler |
|---|---|---|---|---|
| | a rising edge of trigger input comes. | ETIFP is selected. | | filter. |

**Figure 24-106. Event mode**




### Single pulse mode

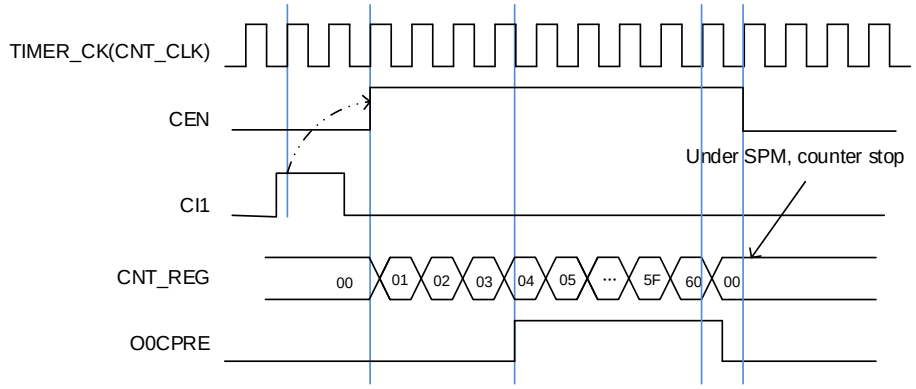
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. If SPM is set, the counter will be cleared and stopped automatically when the next update event occurs. In order to get a pulse waveform, the TIMERx is configured to PWM mode or compare mode by CHxCOMCTL or MCHxCOMCTL bits.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. Setting the CEN bit to 1 or a trigger signal edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 by software, the counter will be stopped and its value will be held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the active edge of trigger which sets the CEN bit to 1 will enable the counter. However, there exists several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE/ MOxCPRE signals will change to, as the compare match event occurs without taking the comparison result into account.

Single pulse mode is also applicable to composite PWM mode (CHxCPWMEN = 1’b1 and CHxMS[2:0] = 3’b000).

**Figure 24-107. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60**




## Delayable single pulse mode

Delayable single pulse mode is enabled by setting CHxCOMCTL[3:0]/ MCHxCOMCTL[3:0] in TIMERx_CHCTLx/ TIMERx_MCHCTLx registers. In this mode, the pulse width of OxCPRE/ MOxCPRE signal is determined by the TIMERx_CAR register.

Once the timer is set to the delayable single pulse mode, the following configuration is required:

- TIMERx need to work in slave mode and TSCFG7[4:0] != 5'b00000 in SYSCFG_TIMERxCFG(x=0,7);
- The CHxCOMCTL[3:0]/ MCHxCOMCTL[3:0] bit-field is setting to 4'b1000 (delayable single pulse mode 0) or 4'b1001 (delayable single pulse mode 1).

In delayable SPM mode 0. The behavior of OxCPRE/ MOxCPRE is performed as in PWM mode 0. When counting up, the OxCPRE/ MOxCPRE is active. When a trigger event occurs, the OxCPRE/ MOxCPRE is inactive. The OxCPRE/ MOxCPRE is active again at the next update event; When counting down, the OxCPRE/ MOxCPRE is inactive, when a trigger event occurs, the OxCPRE/ MOxCPRE is active. The OxCPRE/ MOxCPRE is inactive again at the next update event.

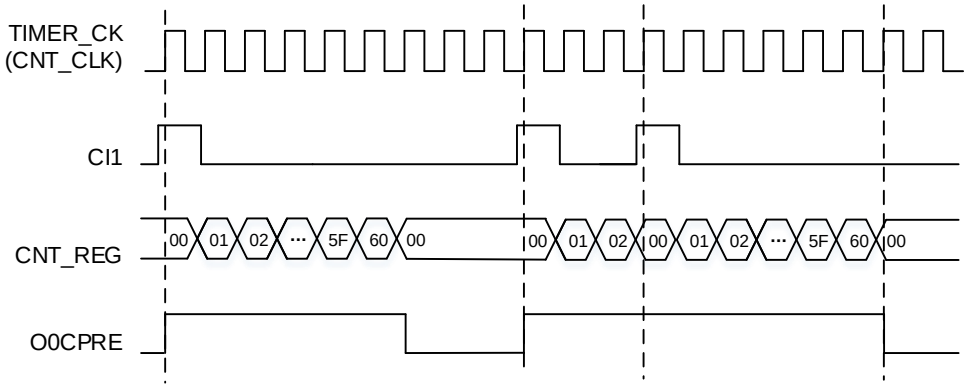
In delayable mode 1. The behavior of OxCPRE/ MOxCPRE is performed as in PWM mode 1. When counting up, the OxCPRE/ MOxCPRE is inactive, when a trigger event occurs, the OxCPRE/ MOxCPRE is active. The OxCPRE/ MOxCPRE is inactive again at the next update event; When counting down, the OxCPRE/ MOxCPRE is active. When a trigger event occurs, the OxCPRE/ MOxCPRE is inactive. The OxCPRE/ MOxCPRE is active again at the next update event.

**Note**:

3) The center-aligned counting mode cannot be used in this mode and the CAM[1:0] = 2'b00( in TIMERx_CTL0 register);
4) When counter counting up (DIR = 0 in TIMERx_CTL0 register), the value of TIMERx_CHxCV/ TIMERx_MCHxCV should be set to 0; When counting down (DIR =1 in TIMERx_CTL0 register), the the value of TIMERx_CHxCV/ TIMERx_MCHxCV should be

greater than or equal to the value of TIMERx_CAR register.

**Figure 24-108. delayable single pulse mode TIMERx_CHxCV=0x00, TIMERx_CAR=0x60**




### Timers interconnection

Please refer to ***Advanced timer (TIMERx, x=0, 7)Timers interconnection***.

### Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx_DMATB, then DMA will access the TIMERx_DMATB. In fact, register TIMERx_DMATB is only a buffer; timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0(1 transfer), then the timer's DMA request is finished. While if TIMERx_DMATC is not 0, such as 3(4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8, DMATA+0xc at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### UPIF bit backup

The UPIF bit backup function is enabled by setting UPIFBUEN in the TIMERx_CTL0 register. The UPIF and UPIFBU bits are fully synchronized and without latency.

By using this function, the UPIF bit in the TIMERx_INTF register will be backuped to the UPIFBU bit in the TIMERx_CNT register. This can avoid conflicts when reading the counter and interrupt processing.

### Timer debug mode

When the Cortex®-M7 halted, and the TIMERx_HOLD configuration bit in DBG_CTL1 register set to 1, the TIMERx counter stops.

### 24.3.5. Register definition (TIMERx, x=14,40,41,42,43,44)

TIMER14 base address: 0x4001 4000

TIMER40 base address: 0x4001 D000

TIMER41 base address: 0x4001 D400

TIMER42 base address: 0x4001 D800

TIMER43 base address: 0x4001 DC00

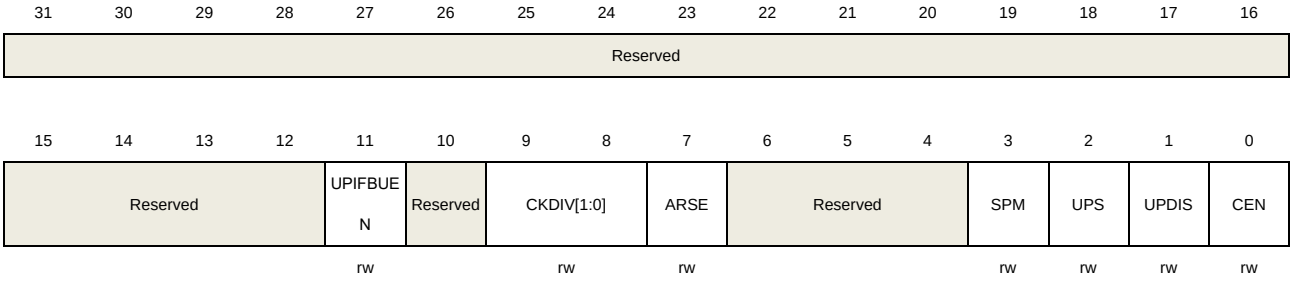
TIMER44 base address: 0x4001 F000

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



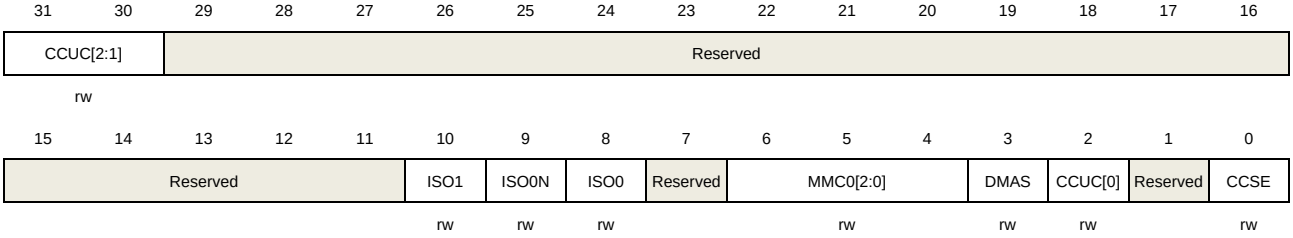

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | UPIFBUEN | UPIF bit backup enable<br>0: Backup disable. UPIF bit is not backuped to UPIFBU bit in TIMERx_CNT register.<br>1: Backup enabled. UPIF bit is backuped to UPIFBU bit in TIMERx_CNT register. |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division ratio between CK_TIMER (the timer clock) and DTS (the dead time and sampling clock) which is used for the dead time generator and the digital filter.<br>00: $f_{DTS}$= $f_{CK_TIMER}$<br>01: $f_{DTS}$= $f_{CK_TIMER}$ /2<br>10: $f_{DTS}$= $f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |

| | | |
|---|---|---|
| 6:4 | Reserved | Must be kept at reset value. |
| 3 | SPM | Single pulse mode<br>0: Single pulse mode is disabled. Counter continues after an update event.<br>1: Single pulse mode is enabled. The CEN bit is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: Any of the following events generates an update interrupt or a DMA request:<br>– The UPG bit is set.<br>– The counter generates an overflow event.<br>– The slave mode controller generates an update event.<br>1: Only counter overflow generates an update interrupt or a DMA request. |
| 1 | UPDIS | Update disable<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. The update event is generated and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set.<br>– The counter generates an overflow event.<br>– The slave mode controller generates an update event.<br>1: Update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or the slave mode controller generates a hardware reset event. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock mode, pause mode or decoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | CCUC[2:1] | Commutation control shadow register update control<br>Refer to CCUC [0] description. |
| 29:11 | Reserved | Must be kept at reset value. |
| 10 | ISO1 | Idle state of channel 1 output<br>Refer to ISO0 bit |
| 9 | ISO0N | Idle state of multi mode channel 0 complementary output<br>0: When POEN bit is reset, MCH0_O is set low.<br>1: When POEN bit is reset, MCH0_O is set high.<br>This bit can be modified only when PROT[1:0] bits in TIMERx_CCHP register is 00. |
| 8 | ISO0 | Idle state of channel 0 output<br>0: When POEN bit is reset, CH0_O is set low.<br>1: When POEN bit is reset, CH0_O is set high.<br>The CH0_O output changes after a dead time if MCH0_O is implemented. This bit can be modified only when PROT[1:0] bits in TIMERx_CCHP register is 00. |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | MMC0[2:0] | Master mode control 0<br>These bits control the selection of TRGO0 signal, which is sent by master timer to slave timer for synchronization function.<br>000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO0 pulse occurs. And in the latter case, the signal on TRGO0 is delayed compared to the actual reset.<br>001: Enable. This mode is used to start several timers at the same time or control a slave timer to be enabled in a period. In this mode, the master mode controller selects the counter enable signal as TRGO0. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO0 output, except if the master-slave mode is selected.<br>010: Update. In this mode, the master mode controller selects the update event as TRGO0.<br>011: Capture/compare pulse. In this mode, the master mode controller generates a TRGO0 pulse when a capture or a compare match occurs in channel 0.<br>100: Compare. In this mode, the master mode controller selects the O0CPRE signal as TRGO0.<br>101: Compare. In this mode, the master mode controller selects the O1CPRE signal as TRGO0.<br>110: Reserved.<br>111: Reserved. |
| 3 | DMAS | DMA request source selection<br>0: DMA request of CHx/MCHx is sent when capture/compare event occurs. |

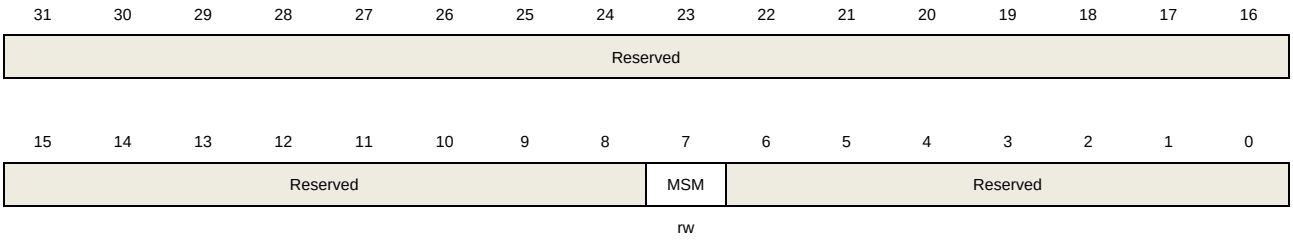
| | | |
|---|---|---|
| | | 1: DMA request of channel CHx/MCHx is sent when update event occurs. |
| 2 | CCUC[0] | Commutation control shadow register update control<br>The CCUC[2:1] and CCUC[0] field are used to control the commutation control shadow register update. When the commutation control shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are enabled (CCSE=1), the update control of the shadow registers with the CCUC[2:0] bit-field are shown as below:<br>000: The shadow registers update when CMTG bit is set.<br>001: The shadow registers update when CMTG bit is set or a rising edge of TRGI occurs.<br>100: The shadow registers update when the counter generates an overflow event.<br>Others: Reserved<br>When a channel does not have a complementary output, this bit has no effect.<br>**Note**: When CCUC[2:0] bit-field are set to 100, the update of the shadow registers also considers the value the CCUSEL bit in the TIMERx_CFG register. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | CCSE | Commutation control shadow enable<br>0: The shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are disabled.<br>1: The shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are enabled.<br>After these bits have been written, they are updated when commutation event comes.<br>When a channel does not have a complementary output, this bit has no effect. |

### Slave mode configuration register (TIMERx_SMCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | MSM | Master-slave mode<br>This bit can be used to synchronize the selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected. |

| | | |
|---|---|---|
| | | 0: Master-slave mode disabled<br>1: Master-slave mode enabled |
| 6:0 | Reserved | Must be kept at reset value. |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CH1COMADDIE | CH0COMADDIE | Reserved | | | MCH0DEN | Reserved | | | MCH0IE | Reserved | | | |
| | | rw | rw | | | | rw | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TRGDEN | CMTDEN | Reserved | | CH1DEN | CH0DEN | UPDEN | BRKIE | TRGIE | CMTIE | Reserved | | CH1IE | CH0IE | UPIE |
| | rw | rw | | | rw | rw | rw | rw | rw | rw | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | CH1COMADDIE | Channel 1 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 28 | CH0COMADDIE | Channel 0 additional compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used in composite PWM mode. |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | MCH0DEN | Multi mode channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH0SEL[1:0] = 2b'00). |
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | MCH0IE | Multi mode channel 0 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH0SEL[1:0] = 2b'00). |

| | | |
|---|---|---|
| 19:15 | Reserved | Must be kept at reset value. |
| 14 | TRGDEN | Trigger DMA request enable<br>0: Disabled<br>1: Enabled |
| 13 | CMTDEN | Commutation DMA request enable<br>0: Disabled<br>1: Enabled |
| 12:11 | Reserved | Must be kept at reset value. |
| 10 | CH1DEN | Channel 1 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Disabled<br>1: Enabled |
| 7 | BRKIE | Break interrupt enable<br>0: Disabled<br>1: Enabled |
| 6 | TRGIE | Trigger interrupt enable<br>0: Disabled<br>1: Enabled |
| 5 | CMTIE | Commutation interrupt enable<br>0: Disabled<br>1: Enabled |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | CH1IE | Channel 1 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 0 | UPIE | Update interrupt enable<br>0: Disabled<br>1: Enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CH1COMADDIF | CH0COMADDIF | Reserved | | | MCH0OF | Reserved | | | MCH0IF | Reserved | | | |
| | | rc_w0 | rc_w0 | | | | rc_w0 | | | | rc_w0 | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | CH1OF | CH0OF | Reserved | BRK0IF | TRGIF | CMTIF | Reserved | | CH1IF | CH0IF | UPIF |
| | | | | | rc_w0 | rc_w0 | | rc_w0 | rc_w0 | rc_w0 | | | rc_w0 | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | CH1COMADDIF | Channel 1 additional compare interrupt flag.<br>Refer to CH0COMADDIF description. |
| 28 | CH0COMADDIF | Channel 0 additional compare interrupt flag.<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in output mode, this flag is set when a compare event occurs.<br>0: No channel 0 output compare interrupt occurred<br>1: Channel 0 output compare interrupt occurred<br>**Note:** This flag just used in composite PWM mode. |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | MCH0OF | Multi mode channel 0 over capture flag<br>When multi mode channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while MCH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | MCH0IF | Multi mode channel 0 capture/compare interrupt flag<br>This flag is set by hardware and cleared by software.<br>If multi mode channel 0 is in input mode, this flag is set when a capture event occurs.<br>If multi mode channel 0 is in output mode, this flag is set when a compare event occurs.<br>If multi mode channel 0 is set to input mode, this bit will be reset by reading TIMERx_MCH0CV.<br>0: No multi mode channel 0 capture/compare interrupt occurred<br>1: Multi mode channel 0 capture/compare interrupt occurred |

| | | |
|---|---|---|
| 19:11 | Reserved | Must be kept at reset value. |
| 10 | CH1OF | Channel 1 over capture flag<br>Refer to CH0OF description |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8 | Reserved | Must be kept at reset value. |
| 7 | BRK0IF | BREAK0 interrupt flag<br>This flag is set by hardware when the break input is active, and cleared by software if the BREAK0 input is not at active level.<br>0: No active level on BREAK0 input has been detected.<br>1: An active level on BREAK0 input has been detected. |
| 6 | TRGIF | Trigger interrupt flag<br>This flag is set by hardware on trigger event and cleared by software.<br>When the slave mode controller is enabled in all modes but pause mode, an active edge of trigger input generates a trigger event. When the slave mode controller is enabled in pause mode, either edge of the trigger input can generate a trigger event.<br>0: No trigger event occurred<br>1: Trigger interrupt occurred |
| 5 | CMTIF | Channel commutation interrupt flag<br>This flag is set by hardware when the commutation event of channel occurs, and cleared by software.<br>0: No channel commutation interrupt occurred<br>1: Channel commutation interrupt occurred |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | CH1IF | Channel 1 capture/compare interrupt flag<br>Refer to CH0IF description |
| 1 | CH0IF | Channel 0 capture/compare interrupt flag<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in input mode, this flag is set when a capture event occurs. If channel 0 is in output mode, this flag is set when a compare event occurs.<br>If channel 0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV.<br>0: No channel 0 interrupt occurred<br>1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware when an update event occurs and cleared by software. |

0: No update interrupt occurred

1: Update interrupt occurred

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CH1COMADDG | CH0COMADDG | Reserved | | | | | | | MCH0G | Reserved | | | |
| | | w | w | | | | | | | | w | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BRK0G | TRGG | CMTG | Reserved | | CH1G | CH0G | UPG |
| | | | | | | | | w | w | w | | | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | CH1COMADDG | Channel 1 additional compare event generation.<br>Refer to CH0COMADDG description. |
| 28 | CH0COMADDG | Channel 0 additional compare event generation.<br>This bit is set by software to generate a compare event in channel 0 additional, it is automatically cleared by hardware.<br>When this bit is set, the CH0COMADDIF flag will be set, and the corresponding interrupt will be sent if enabled.<br>0: No generate a channel 0 additional compare event<br>1: Generate a channel 0 additional compare event<br>**Note:** This bit just used in composite PWM mode. |
| 27:21 | Reserved | Must be kept at reset value. |
| 20 | MCH0G | Multi mode channel 0 capture or compare event generation.<br>This bit is set by software to generate a capture or compare event in multi mode channel 0, it is automatically cleared by hardware. When this bit is set, the MCH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if multi mode channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_MCH0CV register, and the MCH0OF flag is set if the MCH0IF flag has been set.<br>0: No generate a multi mode channel 0 capture or compare event<br>1: Generate a multi mode channel 0 capture or compare event |
| 19:8 | Reserved | Must be kept at reset value. |
| 7 | BRK0G | BREAK0 event generation |

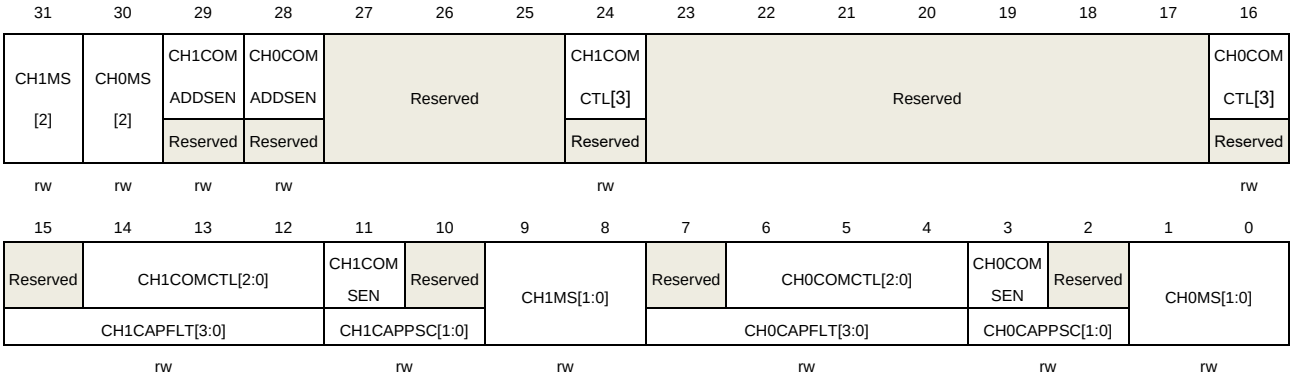
| | | |
|---|---|---|
| | | This bit is set by software to generate an event and cleared by hardware automatically. When this bit is set, the POEN bit will be cleared and BRK0IF flag will be set, related interrupt can occur if enabled.<br>0: No generate a BREAK0 event<br>1: Generate a BREAK0 event |
| 6 | TRGG | Trigger event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register will be set, related interrupt or DMA transfer can occur if enabled.<br>0: No generate a trigger event<br>1: Generate a trigger event |
| 5 | CMTG | Channel commutation event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, MCHxEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).<br>0: No affect<br>1: Generate channel commutation update event |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | CH1G | Channel 1 capture or compare event generation<br>Refer to CH0G description |
| 1 | CH0G | Channel 0 capture or compare event generation<br>This bit is set by software to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag has been set.<br>0: No generate a channel 0 capture or compare event<br>1: Generate a channel 0 capture or compare event |
| 0 | UPG | Update event generation<br>This bit can be set by software, and automatically cleared by hardware. When this bit is set, the counter is cleared if the up counting mode is selected. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH1MS[2] | Channel 1 I/O mode selection<br>Refer to CH1MS[1:0]description |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Refer to CH0MS[1:0] description |
| 29 | CH1COMADDSEN | Channel 1 additional compare output shadow enable<br>Refer to CH0COMADDSEN description. |
| 28 | CH0COMADDSEN | Channel 0 additional compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0COMV_ADD register which updates at each update event will be enabled.<br>0: Channel 0 additional compare output shadow disabled<br>1: Channel 0 additional compare output shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000. |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | CH1COMCTL[3] | Channel 1 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 23:17 | Reserved | Must be kept at reset value. |
| 16 | CH0COMCTL[3] | Channel 0 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | CH1COMCTL[2:0] | Channel 1 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 11 | CH1COMSEN | Channel 1 output compare shadow enable<br>Refer to CH0COMSEN description |

| | | |
|---|---|---|
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CH1MS[1:0] | Channel 1 mode selection<br>This bit-field specifies the direction of the channel and the input signal selection. The CH1MS[2:0] bit-field is writable only when the channel is not active (When MCH1MSEL[1:0] = 2b'00, the CH1EN bit in TIMERx_CHCTL2 register is reset; when MCH1MSEL[1:0] = 2b'11, the CH1EN and MCH1EN bits in TIMERx_CHCTL2 register are reset).<br>000: Channel 1 is configured as output.<br>001: Channel 1 is configured as input, IS1 is connected to CI1FE1.<br>010: Channel 1 is configured as input, IS1 is connected to CI0FE1.<br>011: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=14,40,41,42,43,44) register).<br>100~111: Reserved. |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>The CH0COMCTL[3] and CH0COMCTL[2:0] bit-field control the behavior of O0CPRE which drives CH0_O. The active level of O0CPRE is high, while the active level of CH0_O depends on CH0P bit.<br>**Note**: When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, This bit-field controls the behavior of O0CPRE which drives CH0_O and MCH0_O. The active level of O0CPRE is high, while the active level of CH0_O and MCH0_O depends on CH0P and MCH0P bits.<br>0000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>0010: Clear the channel output on match. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.<br>0011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.<br>0100: Force low. O0CPRE is forced low level.<br>0101: Force high. O0CPRE is forced high level.<br>0110: PWM mode 0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV, otherwise it is inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV, otherwise it is active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of O0CPRE is performed as in PWM mode 0. When counting up, the O0CPRE is active. When a trigger event occurs, |

| | | |
|---|---|---|
| | | the O0CPRE is inactive. The O0CPRE is active again at the next update event; When counting down, the O0CPRE is inactive, when a trigger event occurs, the O0CPRE is active. The O0CPRE is inactive again at the next update event.<br>1001: Delayable SPM mode 1. The behavior of O0CPRE is performed as in PWM mode 1. When counting up, the O0CPRE is inactive, when a trigger event occurs, the O0CPRE is active. The O0CPRE is inactive again at the next update event; When counting down, the O0CPRE is active. When a trigger event occurs, the O0CPRE is inactive. The O0CPRE is active again at the next update event.<br>1010~1111: Reserved.<br>**Note**: In the composite PWM mode (CH0CPWMEN = 1'b1 and CH0MS = 3'b000), the PWM signal output in channel 0 is composited by TIMERx_CH0CV and TIMERx_CH0COMV_ADD. Please refer to ***Composite PWM mode*** for more details.<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from "Timing" mode to "PWM" mode or the result of the comparison changes.<br>When the outputs of CH0 and MCH0 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000 (compare mode). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register which updates at each update event will be enabled.<br>0: Channel 0 output compare shadow disabled<br>1: Channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. The CH0MS[2:0] bit-field is writable only when the channel is not active (When MCH0MSEL[1:0] = 2b'00, the CH1EN bit in TIMERx_CHCTL2 register is reset; when MCH0MSEL[1:0] = 2b'11, the CH0EN and MCH0EN bits in TIMERx_CHCTL2 register are reset).<br>000: Channel 0 is configured as output.<br>001: Channel 0 is configured as input, IS0 is connected to CI0FE0.<br>010: Channel 0 is configured as input, IS0 is connected to CI1FE0.<br>011: Channel 0 is configured as input, IS0 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=14,40,41,42,43,44) register). |

100: Channel 0 is configured as input, IS0 is connected to MCI0FE0.
101~111: Reserved.

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CH1MS[2] | Channel 1 I/O mode selection<br>Same as output compare mode. |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Same as output compare mode. |
| 29:16 | Reserved | Must be kept at reset value. |
| 15:12 | CH1CAPFLT[3:0] | Channel 1 input capture filter control<br>Refer to CH0CAPFLT description. |
| 11:10 | CH1CAPPSC[1:0] | Channel 1 input capture prescaler<br>Refer to CH0CAPPSC description. |
| 9:8 | CH1MS[1:0] | Channel 1 I/O mode selection<br>Same as output compare mode. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges. |

| | | |
|---|---|---|
| | | 10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>Same as output compare mode. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | MCH1P | Reserved | CH1P | CH1EN | MCH0P | MCH0EN | CH0P | CH0EN |
| | | | | | | | | rw | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | MCH1P | Multi mode channel 1 output polarity<br>Refer to MCH0P description. |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CH1P | Channel 1 capture/compare polarity<br>Refer to CH0P description. |
| 4 | CH1EN | Channel 1 capture/compare enable<br>Refer to CH0EN description. |
| 3 | MCH0P | Multi mode channel 0 output polarity<br>When Multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, this bit specifies the MCH0_O output signal polarity.<br>0: Multi mode channel 0 output active high<br>1: Multi mode channel 0 output active low<br>When CH0 is configured in input mode, in conjunction with CH0P, this bit is used to define the polarity of CH0.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 2 | MCH0EN | Multi mode channel 0 capture/compare enable<br>When multi mode channel 0 is configured in output mode, setting this bit enables MCH0_O signal in active state. When multi mode channel 0 is configured in input mode, setting this bit enables the capture event in multi mode channel 0.<br>0: Multi mode channel 0 disabled |

| | | |
|---|---|---|
| | | 1: Multi mode channel 0 enabled |
| 1 | CH0P | Channel 0 capture/compare polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, these bits specifie the channel 0 input signal's polarity. [MCH0P, CH0P] will select the active trigger or capture polarity for channel 0 input signals.<br>00: Channel 0 input signal's rising edge is the active signal for capture or trigger operation in slave mode. And channel 0 input signal will not be inverted.<br>01: Channel 0 input signal's falling edge is the active signal for capture or trigger operation in slave mode. And channel 0 input signal will be inverted.<br>10: Reserved.<br>11: Noninverted/both channel 0 input signal's edges.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel 0.<br>0: Channel 0 disabled<br>1: Channel 0 enabled |

### Counter register (TIMERx_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UPIFBU | Reserved | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register, and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |

| | | |
|---|---|---|
| 30:16 | Reserved | Must be kept at reset value. |
| 15:0 | CNT[15:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

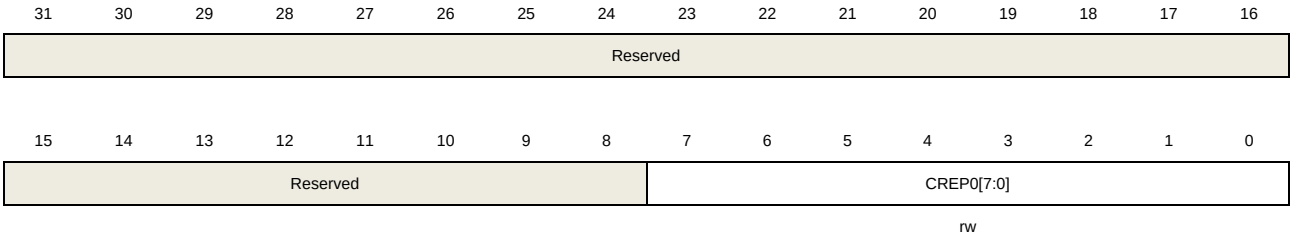
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-field specifies the auto reload value of the counter. |

### Counter repetition register 0 (TIMERx_CREP0)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



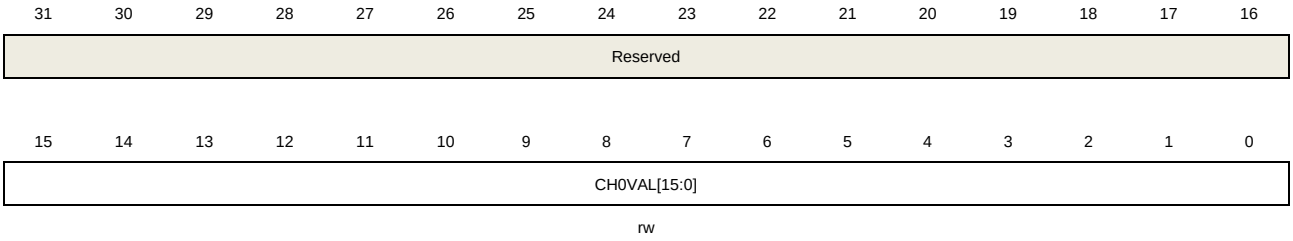

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | CREP0[7:0] | Counter repetition value 0<br>This bit-field specifies the update event generation rate. Each time the repetition counter counts down to zero, an update event will be generated. The update rate of the shadow registers is also affected by this bit-field when these shadow registers are enabled.<br>**Note**: This bit-field just used with CREPSEL =0 (in TIMERx_CFG register). |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



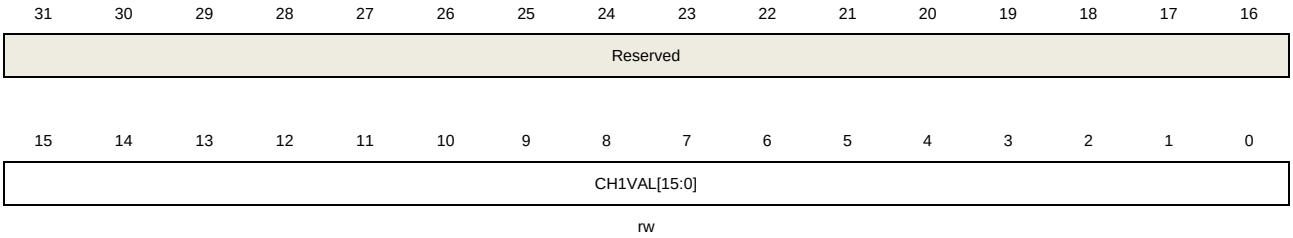

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH0VAL[15:0] | Capture/compare value of channel 0<br>When channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Channel 1 capture/compare value register (TIMERx_CH1CV)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



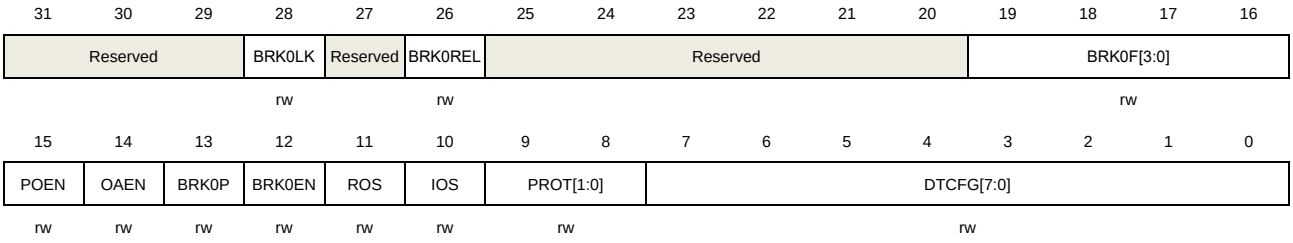

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH1VAL[15:0] | Capture/compare value of channel 1<br>When channel 1 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | BRK0LK | BREAK0 input locked<br>0: BREAK0 input in input mode<br>1: BREAK0 input in locked mode<br>When the BRK0LK is set to 1, the BREAK0 input is configured in open drain output mode.<br>Any active BREAK0 event asserts a low logic level on the Break input to indicate an internal break event to external devices. |

This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

**Note**: Every write operation to this bit needs a delay of 1 APB clock to active.

27 Reserved Must be kept at reset value.

26 BRK0REL BREAK0 input released

This bit is cleared by hardware when the BREAK0 input is invalid.

0: BREAK0 input is unreleased

1: BREAK0 input is released

The locked output control (open drain mode in Hi-z state) is released by setting this bit with software. And when the fault is disappeared, this bit will reset by hardware.

**Note**: Every write operation to this bit needs a delay of 1 APB clock to active.

25:20 Reserved Must be kept at reset value.

19:16 BRK0F[3:0] BREAK0 input signal filter

An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample BREAK0 input signal and the length of the digital filter applied to BREAK0.

0000: Filter disabled. BREAK0 act asynchronously, N=1

0001: $f_{SAMP} = f_{CK_TIMER}$, N=2

0010: $f_{SAMP} = f_{CK_TIMER}$, N=4

0011: $f_{SAMP} = f_{CK_TIMER}$, N=8

0100: $f_{SAMP} = f_{DTS}/2$, N=6

0101: $f_{SAMP} = f_{DTS}/2$, N=8

0110: $f_{SAMP} = f_{DTS}/4$, N=6

0111: $f_{SAMP} = f_{DTS}/4$, N=8

1000: $f_{SAMP} = f_{DTS}/8$, N=6

1001: $f_{SAMP} = f_{DTS}/8$, N=8

1010: $f_{SAMP} = f_{DTS}/16$, N=5

1011: $f_{SAMP} = f_{DTS}/16$, N=6

1100: $f_{SAMP} = f_{DTS}/16$, N=8

1101: $f_{SAMP} = f_{DTS}/32$, N=5

1110: $f_{SAMP} = f_{DTS}/32$, N=6

1111: $f_{SAMP} = f_{DTS}/32$, N=8

This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

15 POEN Primary output enable

This bit is set by software or automatically set by hardware depending on the OAEN bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and MCHx_O) if the corresponding enable bits (CHxEN, MCHxEN in TIMERx_CHCTL2 register) have been set.

0: Channel outputs are disabled or forced to idle state.

| | | |
|---|---|---|
| | | 1: Channel outputs are enabled. |
| 14 | OAEN | Output automatic enable<br>This bit specifies whether the POEN bit can be set automatically by hardware.<br>0: POEN cannot be set by hardware.<br>1: POEN can be set by hardware automatically at the next update event, if the break input is not active.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 13 | BRK0P | BREAK0 input signal polarity<br>This bit specifies the polarity of the BREAK0 input signal.<br>0: BREAK0 input active low<br>1: BREAK0 input active high<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 12 | BRK0EN | BREAK0 input signal enable<br>This bit can be set to enable the BREAK0 input signal<br>0: BREAK0 input disabled<br>1: BREAK0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2’b11)***.<br>0: “off-state” disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.<br>1: “off-state” enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2’b11)***.<br>0: “off-state” disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CHxEN/CHxNEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is |

10 or 11.

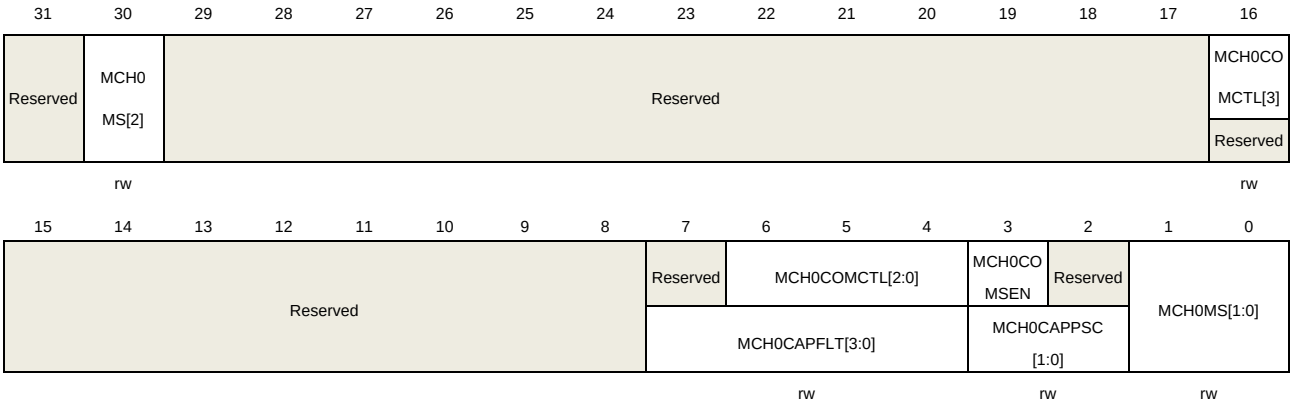
| | | |
|---|---|---|
| 9:8 | PROT[1:0] | Complementary register protect control<br>This bit-field specifies the write protection property of registers.<br>00: Protect disabled. No write protection.<br>01: PROT mode 0. The ISOx/ISOxN bits in TIMERx_CTL1 register, the BRK0EN/ BRK0P/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.<br>10: PROT mode 1. In addition to the registers in PROT mode 0, the CHxP/MCHxP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) the ROS/IOS bits in TIMERx_CCHP register are writing protected.<br>11: PROT mode 2. In addition to the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN/ CHxCOMADDSEN/ MCHxCOMCTL/ MCHxCOMSEN bits in TIMERx_CHCTL0 and TIMERx_MCHCTL0 registers (if the related channel is configured in output) are writing protected.<br>This bit-field can be written only once after the system reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected. |
| 7:0 | DTCFG[7:0] | Dead time configuration<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between the value of DTCFG and the duration of dead-time is as follow:<br>DTCFG[7:5] = 3'b0xx: DT value = DTCFG[7:0] * $t_{DT}$, $t_{DT}$ = $t_{DTS}$.<br>DTCFG[7:5] = 3'b10x: DT value = (64+DTCFG[5:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*2.<br>DTCFG[7:5] = 3'b110: DT value = (32+DTCFG[4:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*8.<br>DTCFG[7:5] = 3'b111: DT value = (32+DTCFG[4:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*16.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### Multi mode channel control register 0 (TIMERx_MCHCTL0)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | MCH0MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH0MS[1:0] description. |
| 29:17 | Reserved | Must be kept at reset value. |
| 16 | MCH0COMCTL[3] | Multi mode channel 0 compare output control.<br>Refer to MCH0COMCTL[2:0] description. |
| 15:7 | Reserved | Must be kept at reset value. |
| 6:4 | MCH0COMCTL[2:0] | Multi mode channel 0 output compare control<br>When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'00, the MCH0COMCTL[3] and MCH0COMCTL[2:0] bit-field control the behavior of MO0CPRE which drives MCH0_O. The active level of MO0CPRE is high, while the active level of MCH0_O depends on MCH0FP[1:0] bits.<br>**Note**: When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, the CH0COMCTL[2:0] bit-field controls the behavior of O0CPRE which drives CH0_O and MCH0_O, while the active level of CH0_O and MCH0_O depends on CH0P and MCH0P bits.<br>0000: Timing mode. The MO0CPRE signal keeps stable, independent of the comparison between the register TIMERx_MCH0CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. MO0CPRE signal is forced high when the counter matches the output compare register TIMERx_MCH0CV.<br>0010: Clear the channel output on match. MO0CPRE signal is forced low when the counter matches the output compare register TIMERx_MCH0CV.<br>0011: Toggle on match. MO0CPRE toggles when the counter matches the output compare register TIMERx_MCH0CV.<br>0100: Force low. MO0CPRE is forced low level.<br>0101: Force high. MO0CPRE is forced high level.<br>0110: PWM mode 0. When counting up, MO0CPRE is active as long as the counter is smaller than TIMERx_MCH0CV, otherwise it is inactive. When counting down, MO0CPRE is inactive as long as the counter is larger than TIMERx_MCH0CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, MO0CPRE is inactive as long as the counter is smaller than TIMERx_MCH0CV, otherwise it is active. When counting down, MO0CPRE is active as long as the counter is larger than TIMERx_MCH0CV, otherwise it is inactive.<br>1000: Delayable SPM mode 0. The behavior of MO0CPRE is performed as in PWM mode 0. When counting up, the MO0CPRE is active. When a trigger event occurs, the MO0CPRE is inactive. The MO0CPRE is active again at the next update event; When counting down, the MO0CPRE is inactive, when a trigger event occurs, the O0CPR MO0CPRE E is active. The MO0CPRE is inactive again at the next update event. |

| | | |
|---|---|---|
| | | 1001: Delayable SPM mode 1. The behavior of MO0CPRE is performed as in PWM mode 1. When counting up, the MO0CPRE is inactive, when a trigger event occurs, the MO0CPRE is active. The MO0CPRE is inactive again at the next update event; When counting down, the MO0CPRE is active. When a trigger event occurs, the MO0CPRE is inactive. The MO0CPRE is active again at the next update event.<br>1010~1111: Reserved.<br>If configured in PWM mode, the MO0CPRE level changes only when the output compare mode switches from “Timing” mode to “PWM” mode or the result of the comparison changes.<br>When the outputs of CH0 and MCH0 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0NMS bit-field is 00(compare mode). |
| 3 | MCH0COMSEN | Multi mode channel 0 output compare shadow enable<br>When this bit is set, the shadow register of TIMERx_MCH0CV register which updates at each update event will be enabled.<br>0: Multi mode channel 0 output compare shadow disabled<br>1: Multi mode channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and MCH0MS bit-field is 00. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | MCH0MS[1:0] | Multi mode channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active (MCH0EN bit in TIMERx_CHCTL2 register is reset).<br>000: Multi mode channel 0 is configured as output.<br>001: Multi mode channel 0 is configured as input, MIS0 is connected to MCI0FEM0.<br>010: Reserved.<br>011: Multi mode channel 0 is configured as input, MIS0 is connected to ITS, this mode is working only if an internal trigger input is selected (through TSCFG15[4:0] bit-field in SYSCFG_TIMERxCFG2(x=14,40,41,42,43,44) register).<br>100: Multi mode channel 0 is configured as input, MIS0 is connected to CI0FEM0.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | MCH0MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH0MS[1:0] description. |

| | | |
|---|---|---|
| 29:8 | Reserved | Must be kept at reset value. |
| 7:4 | MCH0CAPFLT[3:0] | Multi mode channel 0 input capture filter control.<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample MCI0 input signal and the length of the digital filter applied to MCI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | MCH0CAPPSC[1:0] | Multi mode channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when MCH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | MCH0MS[1:0] | Multi mode channel 0 I/O mode selection<br>Same as output compare mode |

### Multi mode channel control register 2 (TIMERx_MCHCTL2)

Address offset: 0x50

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | MCH0FP[1:0] | |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1:0 | MCH0FP[1:0] | Multi mode channel 0 capture/compare free polarity<br>When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'00, these bits specifie the multi mode channel 0 output signal polarity.<br>00: Multi mode channel 0 active high<br>01: Multi mode channel 0 active low<br>10: Reserved.<br>11: Reserved.<br>When multi mode channel 0 is configured in input mode, these bits specifie the multi mode channel 0 input signal's polarity. MCH0FP[1:0] will select the active trigger or capture polarity for multi mode channel 0 input signals.<br>00: Multi mode channel 0 input signal's rising edge is the active signal for capture or trigger operation in slave mode. And multi mode channel 0 input signal will not be inverted.<br>01: Multi mode channel 0 input signal's falling edge is the active signal for capture or trigger operation in slave mode. And multi mode channel 0 input signal will be inverted.<br>10: Reserved.<br>11: Noninverted/both multi mode channel 0 input signal's edges.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |

### Multi mode channel 0 capture/compare value register (TIMERx_MCH0CV)

Address offset: 0x54

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MCH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | MCH0VAL[15:0] | Capture/compare value of multi mode channel 0.<br>When multi mode channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When multi mode channel 0 is configured in output mode, this bit-field contains |

value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.

### Channel 0 additional compare value register (TIMERx_CH0COMV_ADD)

Address offset: 0x64
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH0COMVAL_ADD [15:0] | Additional compare value of channel 0<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event.<br>**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1). |

### Channel 1 additional compare value register (TIMERx_CH1COMV_ADD)

Address offset: 0x68
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH1COMVAL_ADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH1COMVAL_ADD [15:0] | Additional compare value of channel 1<br>When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

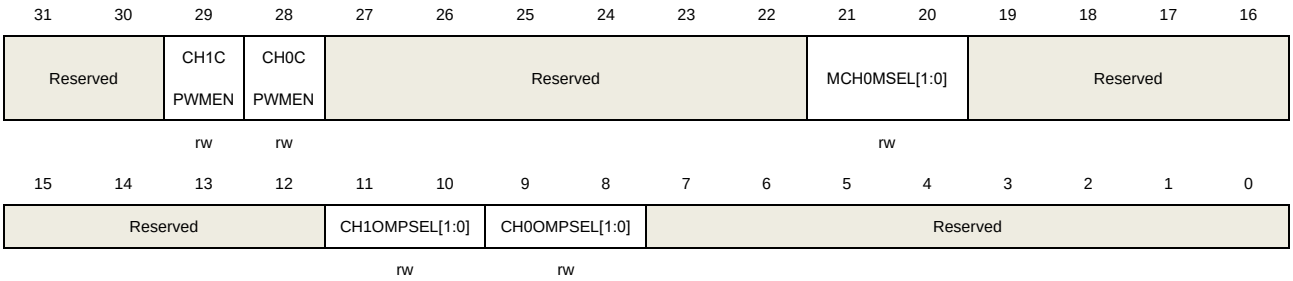
**Note:** This register just used in composite PWM mode(when CH0CPWMEN=1).

### Control register 2 (TIMERx_CTL2)

Address offset: 0x74

Reset value: 0x0030 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | CH1CPWMEN | Channel 1 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 28 | CH0CPWMEN | Channel 0 composite PWM mode enable<br>0: Disabled<br>1: Enabled |
| 27:22 | Reserved | Must be kept at reset value. |
| 21:20 | MCH0MSEL[1:0] | Multi mode channel 0 mode select<br>00: Independent mode, MCH0 is independent of CH0<br>01: Reserved<br>10: Reserved<br>11: Complementary mode, only the CH0 is valid for input, and the outputs of MCH0 and CH0 are complementary |
| 19:12 | Reserved | Must be kept at reset value. |
| 11:10 | CH1OMPSEL[1:0] | Channel 1 output match pulse select<br>When the match events occur, this bit is used to select the output of O1CPRE which drives CH1_O.<br>00: The O1CPRE signal is output normal with the configuration of CH1COMCTL[2:0] bits.<br>01: Only when the counter is counting up, the O1CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Reserved<br>11: Reserved |

| | | |
|---|---|---|
| 9:8 | CH0OMPSEL[1:0] | Channel 0 output match pulse select<br>When the match events occur, this bit is used to select the output of O0CPRE which drives CH0_O.<br>00: The O0CPRE signal is output normal with the configuration of CH0COMCTL[2:0] bits.<br>01: Only when the counter is counting up, the O0CPRE signal is output a pulse when the match events occurs, and the pulse width is one CK_TIMER clock cycle.<br>10: Reserved<br>11: Reserved |
| 7:0 | Reserved | Must be kept at reset value. |

### TIMERx alternate function control register 0 (TIMERx_AFCTL0)

Address offset: 0x8C
Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1P | BRK0CMP0P | Reserved | | | | | | | | BRK0IN0P |
| | | | | | rw | rw | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1EN | BRK0CMP0EN | BRK0HPDFEN | Reserved | | | | | | | BRK0IN0EN |
| | | | | | rw | rw | rw | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | BRK0CMP1P | BREAK0 CMP1 input polarity<br>This bit is used to configure the CMP1 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: CMP1 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP1 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 25 | BRK0CMP0P | BREAK0 CMP0 input polarity<br>This bit is used to configure the CMP0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: CMP0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP0 input signal will be inverted (BRK0P=0, the input signal is active high; |

| | | |
|---|---|---|
| | | BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 24:17 | Reserved | Must be kept at reset value. |
| 16 | BRK0IN0P | BREAK0 BRKIN0 alternate function input polarity<br>This bit is used to configure the BRKIN0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | BRK0CMP1EN | BREAK0 CMP1 enable<br>0: CMP1 input disabled<br>1: CMP1 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 9 | BRK0CMP0EN | BREAK0 CMP0 enable<br>0: CMP0 input disabled<br>1: CMP0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 8 | BRK0HPDFEN | BREAK0 HPDF input(hpdf_break[x], please refer to ***Table 45-2. HPDF internal signal***) enable<br>0: HPDF input disabled<br>1: HPDF input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | BRK0IN0EN | BREAK0 BRKIN0 alternate function input enable<br>0: BRKIN0 alternate function input disabled<br>1: BRKIN0 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### Counter repetition register 1 (TIMERx_CREP1)

Address offset: 0x98

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CREP1[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CREP1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CREP1[31:0] | Counter repetition value 1<br>This bit-field is 32 bits and can be read on the fly.<br>This bit-field specifies the update event generation rate. Each time the repetition counter counts down to zero, an update event will be generated. The update rate of the shadow registers is also affected by this bit-field when these shadow registers are enabled.<br>**Note**: This bit-field just used with CREPSEL =1(in TIMERx_CFG register). |

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0xE0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DMATC[5:0] | | | | | | Reserved | | DMATA[5:0] | | | | | |
| | | rw | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:8 | DMATC[5:0] | DMA transfer count<br>This field defines the times of accessing(R/W) the TIMERx_DMATB register by DMA.<br>6’b000000: transfer 1 time<br>6’b000001: transfer 2 times<br>…<br>6’b100101: transfer 38 times |
| 7:6 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 5:0 | DMATA[5:0] | DMA transfer access start address<br>This field defines the start address of accessing the TIMERx_DMATB register by DMA. When the first access to the TIMERx_DMATB register is done, this bit-field specifies the address just accessed. And then the address of the second access to the TIMERx_DMATB register will be (start address + 0x4).<br>6'b000000: TIMERx_CTL0<br>6'b000001: TIMERx_CTL1<br>…<br>In a word: start address = TIMERx_CTL0 + DMATA*4 |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0xE4

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMATB[31:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address ranges from (start address) to (start address + transfer count * 4) will be accessed.<br>The transfer count is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCUSEL | CREPSEL | CHVSEL | OUTSEL |
| | | | | | | | | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | CCUSEL | Commutation control shadow register update select<br>This bit is valid only when the CCUC[2:0] bit-field are set to 100.<br>0: The shadow registers update when the counter generates an overflow/ underflow event.<br>1: The shadow registers update when the counter generates an overflow/ underflow event and the repetition counter value is zero. |
| 2 | CREPSEL | The counter repetition register select<br>This bit is used to select the counter repetition register.<br>0: The update event rate is depended to TIMERx_CREP0 register<br>1: The update event rate is depended to TIMERx_CREP1 register |
| 1 | CHVSEL | Write CHxVAL register selection bit<br>This bit-field is set and reset by software.<br>1: If the value to be written to the CHxVAL register is the same as the value of CHxVAL register, the write access is ignored.<br>0: No effect. |
| 0 | OUTSEL | The output value selection bit<br>This bit-field is set and reset by software.<br>1: If POEN bit and IOS bit are 0, the output is disabled.<br>0: No effect. |

## 24.4. General level4 timer (TIMERx, x=15,16)

### 24.4.1. Overview

The general level4 timer module (TIMER15/16) is a two-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level4 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level4 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which is suitable for motor control applications.

### 24.4.2. Characteristics

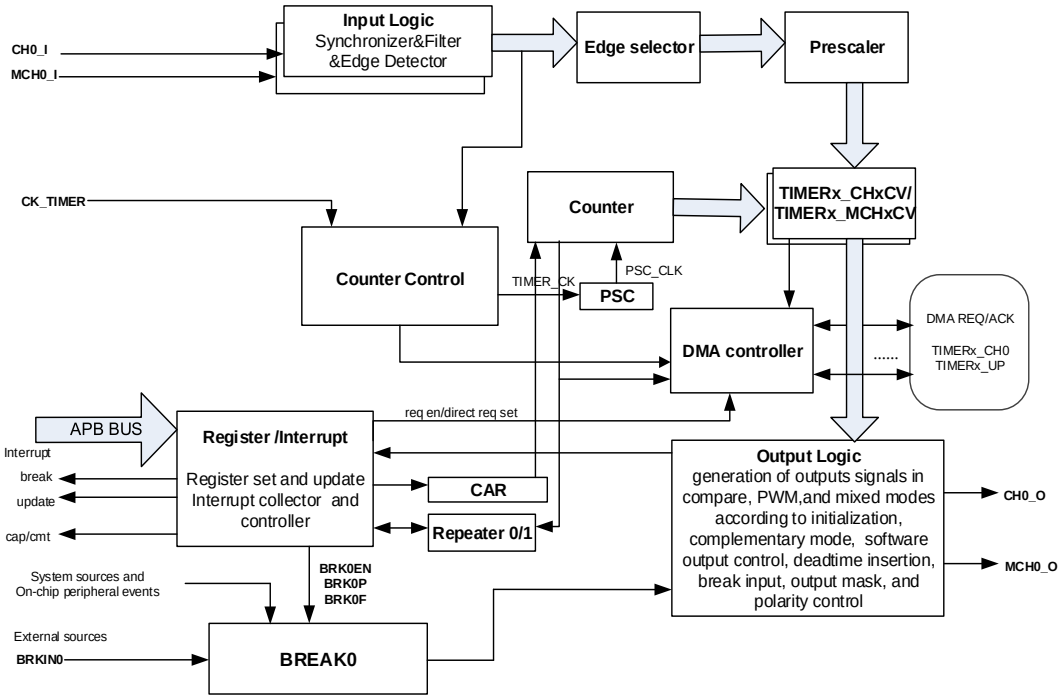
- Total channel num: 2.
- Counter width: 16-bit.
- Source of counter clock: internal clock.
- Counter modes: count up only.
- Programmable prescaler: 16-bit. The factor can be changed on the go.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode, single pulse mode.
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input function: BREAK0.
- Interrupt output or DMA request: update, compare/capture event, and break input.

### 24.4.3. Block diagram

*Figure 24-109. General level4 timer block diagram* provides details of the internal

configuration of the general level4 timer.

**Figure 24-109. General level4 timer block diagram**




## 24.4.4. Function overview

### Clock selection

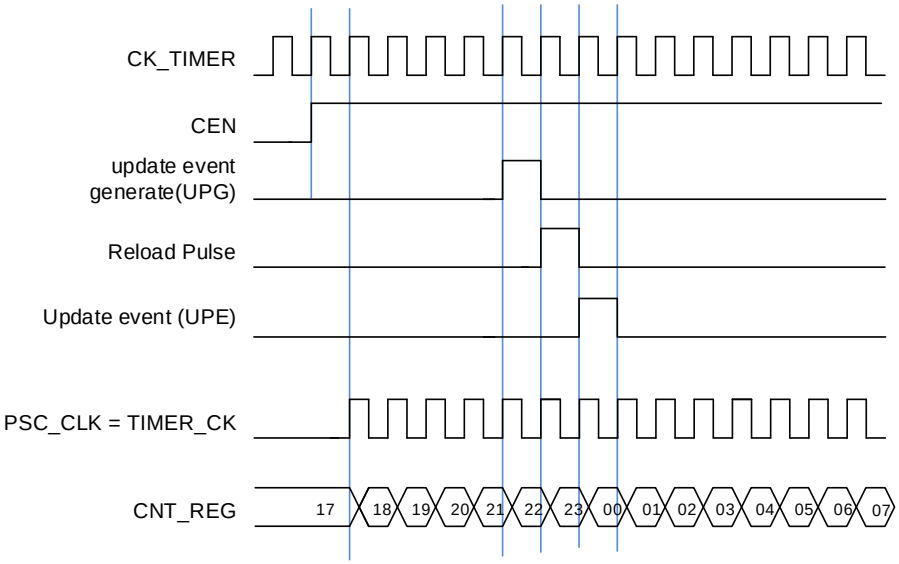
The general level4 TIMER can only being clocked by the CK_TIMER.

- Internal timer clock CK_TIMER which is from module RCU

The general level4 TIMER has only one clock source which is the internal CK_TIMER, used to drive the counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

The TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU

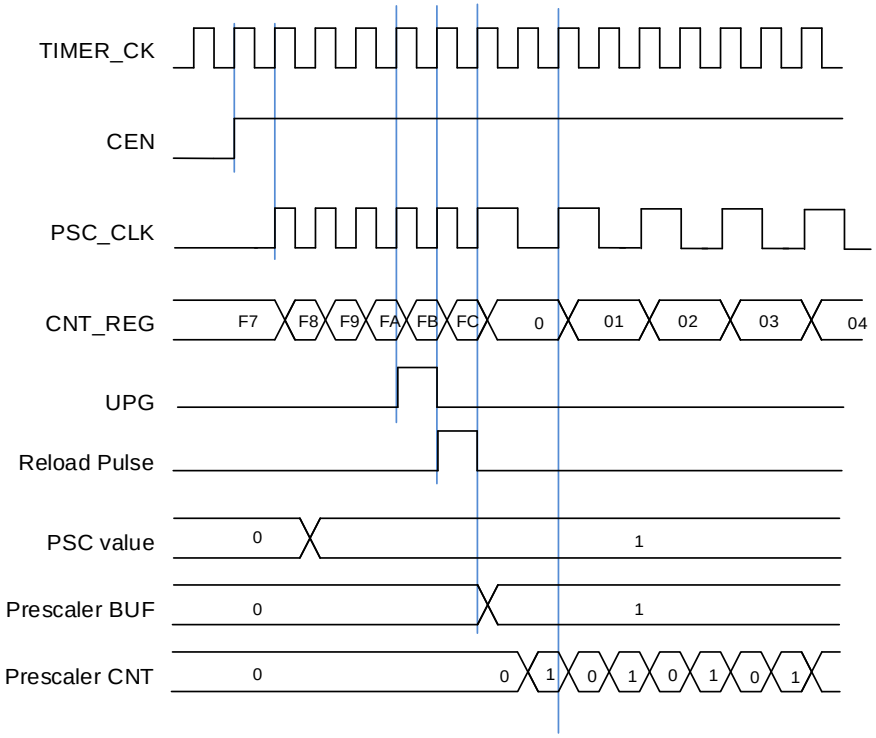
**Figure 24-110. Normal mode, internal clock divided by 1**




## Prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMERx_PSC) which can be changed on the go but is taken into account at the next update event.

**Figure 24-111. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMERx_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after (TIMERx_CREP0/1+1) times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If set the UPDIS bit in TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

*Figure 24-112. Timing diagram of up counting mode, PSC=0/2* and *Figure 24-113. Timing diagram of up counting mode, change TIMERx_CAR on the go* show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

**Figure 24-112. Timing diagram of up counting mode, PSC=0/2**

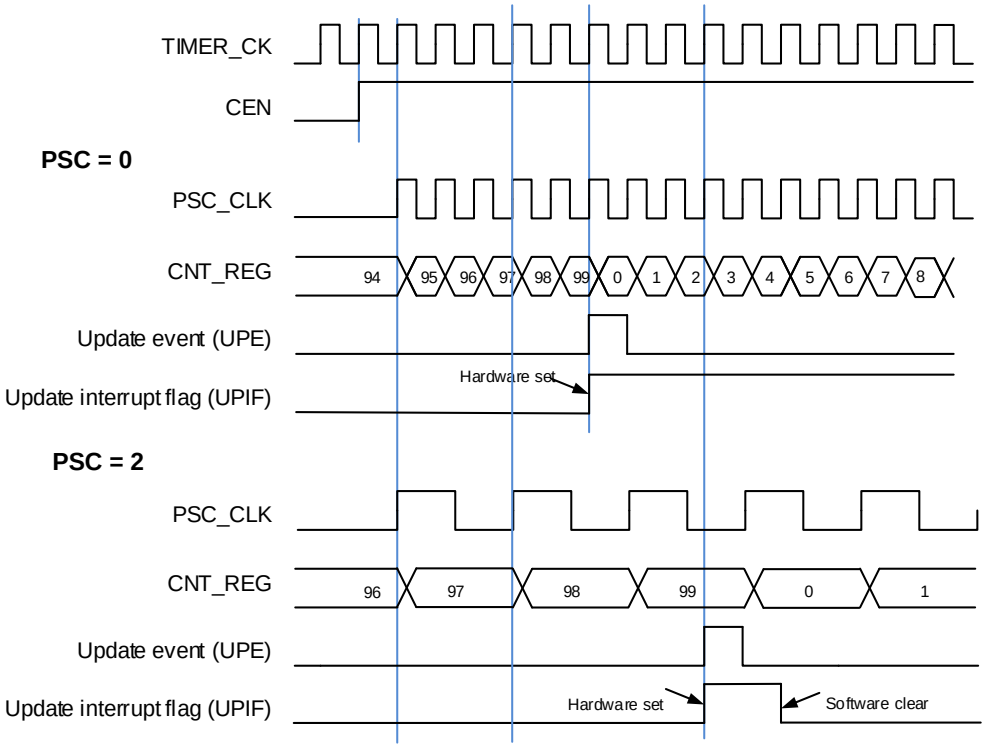
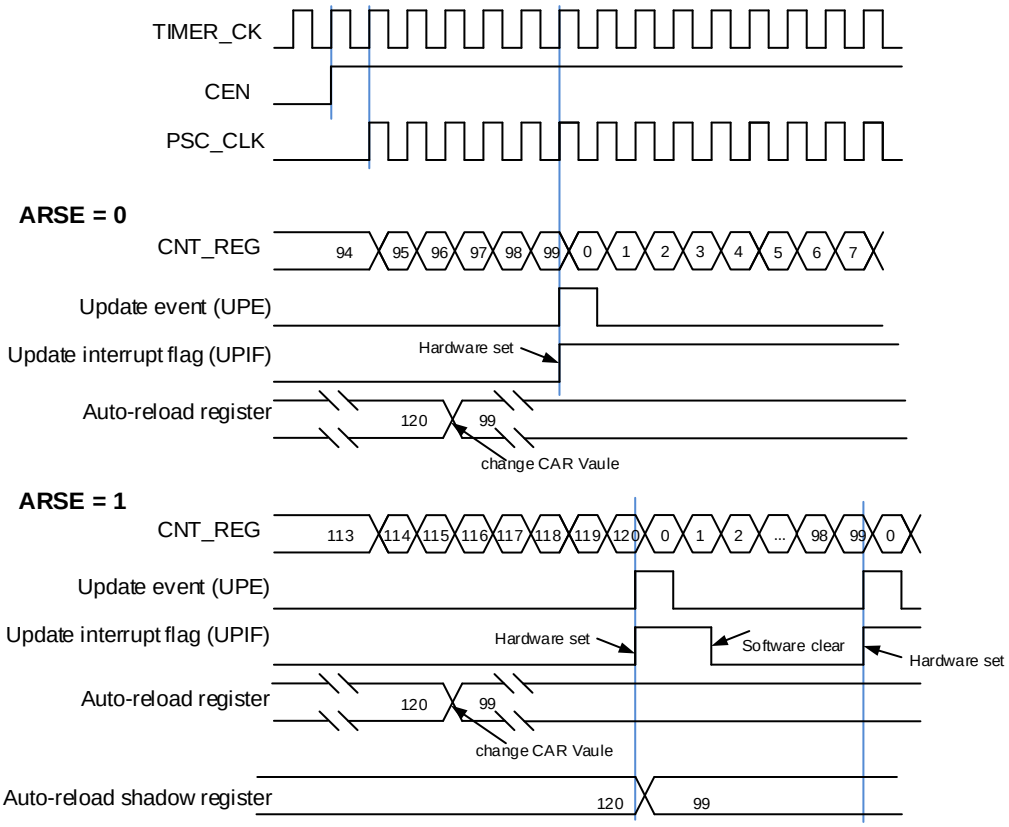


**Figure 24-113. Timing diagram of up counting mode, change TIMERx_CAR on the go**



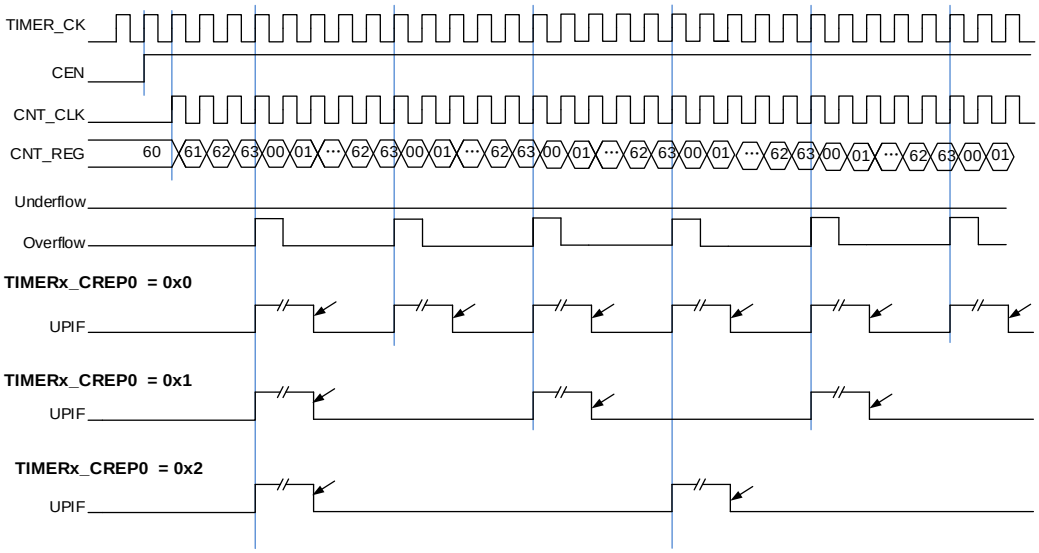

## Counter repetition

The general timer has two repetitions counter TIMERx_CREP0/1, which can be selected by configuring the CPERSEL bit in the TIMERx_CFG register. The CPEP[7:0] bit-field is 8bits, the CPEP[31:0] bit-field is 32bits and can be read on the fly.

Counter repetition is used to generator update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP0/1 in TIMERx_CREP0/1 register. The repetition counter is decremented at each counter overflow in up-counting mode.

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP0/1 in TIMERx_CREP0/1 register and generator an update event.

**Figure 24-114. Repetition timechart for up-counter**




## Capture/compare channels

The general level4 timer has two independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

When the channels are used for input, channel 0 and multi mode channel 0 can perform input capture independently; when the channels are used for comparison output, the channel 0 and multi mode channel 0 can output independent and complementary outputs.

- **Input capture mode**

When MCHxMSEL=2'b00(independent mode), channel x and multi mode channel x can perform input capture independently.

Input capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx_CHxCV/ TIMERx_MCHxCV(x=0) registers, at the same time the CHxIF/ MCHxIF(x=0) bits are set and the channel interrupt is generated if it is enabled when CHxIE/ MCHxIE =1(x=0).

**Figure 24-115. Input capture logic for channel 0**

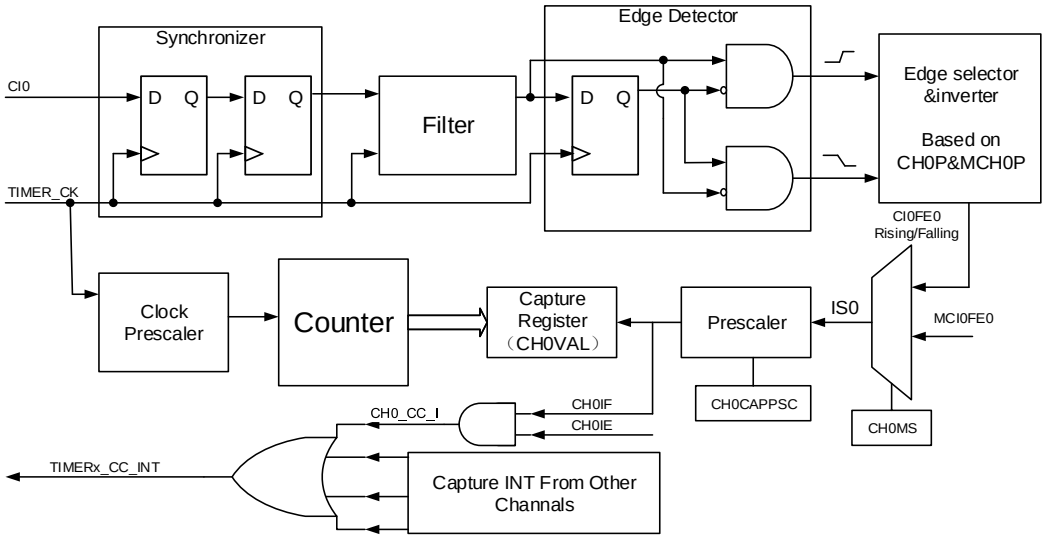
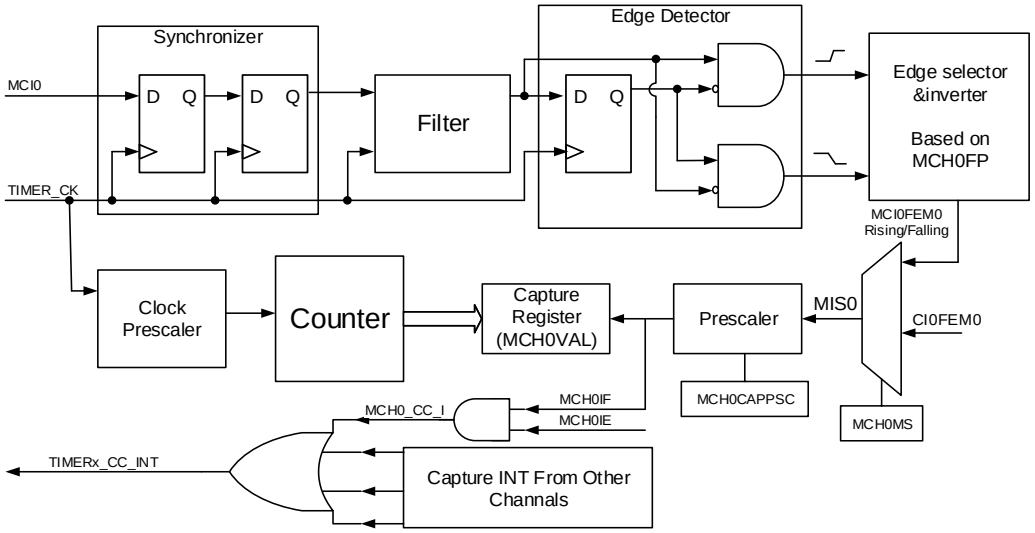



**Figure 24-116. Input capture logic for multi mode channel 0**




The input signals of channelx (CIx/ MCIx) are the TIMERx_CHx/ TIMERx_MCHxCV signal.

First, the input signal of channel (CIx/ MCIx) is synchronized to TIMER_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP/ MCHxP or MCHxFP bits. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS/ MCHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx_CHxCV/ TIMERx_MCHxCV will store the value of counter.

So, the process can be divided into several steps as below:

**Step1**: Filter configuration (CHxCAPFLT bit in TIMERx_CHCTL0 register and MCHxCAPFLT bit in TIMERx_MCHCTL0 register).

Based on the input signal and quality of requested signal, configure compatible CHxCAPFLT or MCHxCAPFLT bit.

**Step2**: Edge selection (CHxP and MCHxP bits in TIMERx_CHCTL2 register, MCHxFP[1:0] bits in TIMERx_MCHCTL2 register).

Rising edge or falling edge, choose one by configuring CHxP and MCHxP bits or MCHxFP[1:0] bits.

**Step3**: Capture source selection (CHxMS bit in TIMERx_CHCTL0 register, MCHxMS bit in TIMERx_MCHCTL0 register).

As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x000 or MCHxMS != 0x000) and TIMERx_CHxCV/ TIMERx_MCHxCV cannot be written any more.

**Step4**: Interrupt enable (CHxIE and CHxDEN bits, MCHxIE and MCHxDEN bits in TIMERx_DMAINTEN).

Enable the related interrupt to get the interrupt and DMA request.

**Step5**: Capture enable (CHxEN and MCHxEN bits in TIMERx_CHCTL2).

**Result**: When the wanted input signal is captured, TIMERx_CHxCV/ TIMERx_MCHxCV will be set by counter's value and CHxIF/ MCHxIF bit is asserted. If the CHxIF/ MCHxIF bit is 1, the CHxOF/ MCHxOF bit will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN bits, MCHxIE and MCHxDEN bits in TIMERx_DMAINTEN.

**Direct generation**: A DMA request or interrupt is generated by setting CHxG directly.

## Output compare mode

*Figure 24-117. Output compare logic (when MCHxMSEL = 2'00, x=0)* and *Figure 24-118. Output compare logic (when MCHxMSEL = 2'11, x=0)* show the logic circuit of output compare mode.

**Figure 24-117. Output compare logic (when MCHxMSEL = 2'00, x=0)**

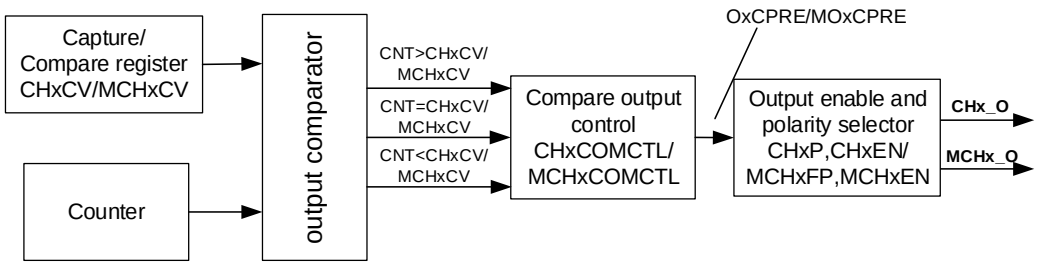
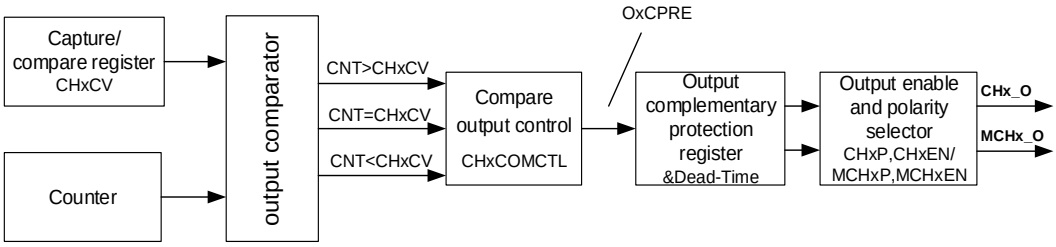


**Figure 24-118. Output compare logic (when MCHxMSEL = 2'11, x=0)**




The relationship between the channel output signal CHx_O/MCHx_O and the OxCPRE/ MOxCPRE signal (more details refer to ***Channel output prepare signal***) is described as blew(the active level of OxCPRE is high and the active level of MOxCPRE is high).

- When MCHxMSEL=2'b00 (in TIMERx_CTL2 register), the MCHx_O output is independent from the CHx_O output. The output level of CHx_O depends on OxCPRE signal, CHxP bit and CHxEN bit (please refer to the TIMERx_CHCTL2 register for more details). The output level of MCHx_O depends on MOxCPRE signal, MCHxFP[1:0] bits and MCHxEN bit (please refer to the TIMERx_MCHCTL2 and TIMERx_CHCTL2 registers for more details). Please refer to ***Figure 24-117. Output compare logic (when MCHxMSEL = 2'00, x=0)***.
- When MCHxMSEL=2'b11, the MCHx_O output is the inverse of the CHx_O output. The output level of CHx_O/MCHx_O depends on OxCPRE signal, CHxP/ MCHxP bits and CHxEN/MCHxEN bits. Please refer to ***Figure 24-118. Output compare logic (when MCHxMSEL = 2'11, x=0)***.

For examples (the MCHx_O output is independent from the CHx_O output):

5) Configure CHxP=0 (the active level of CHx_O is high, the same as OxCPRE), CHxEN=1 (the output of CHx_O is enabled):
   If the output of OxCPRE is active(high) level, the output of CHx_O is active(high) level;
   If the output of OxCPRE is inactive(low) level, the output of CHx_O is active(low) level.

6) Configure MCHxP=1 (the active level of MCHx_O is low, contrary to MOxCPRE), MCHxEN=1 (the output of MCHx_O is enabled):
   If the output of MOxCPRE is active(high) level, the output of MCHx_O is active(low) level;
   If the output of MOxCPRE is inactive(low) level, the output of MCHx_O is active(high) level.

When MCHxMSEL=2'b11 and CHx_O and MCHx_O are output at the same time, the specific outputs of CHx_O and MCHx_O are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx_CCHP register. Please refer to ***Outputs complementary*** for more details.

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the TIMERx_CHxCV/ TIMERx_MCHxCV register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL/ MCHxCOMCTL. When the

counter reaches the value in the TIMERx_CHxCV/ TIMERx_MCHxCV register, the CHxIF/ MCHxIF bit will be set and the channel (n) interrupt is generated if CHxIE/ MCHxIE = 1. And the DMA request will be asserted, if CHxDEN/ MCHxDEN =1.

So, the process can be divided into several steps as below:

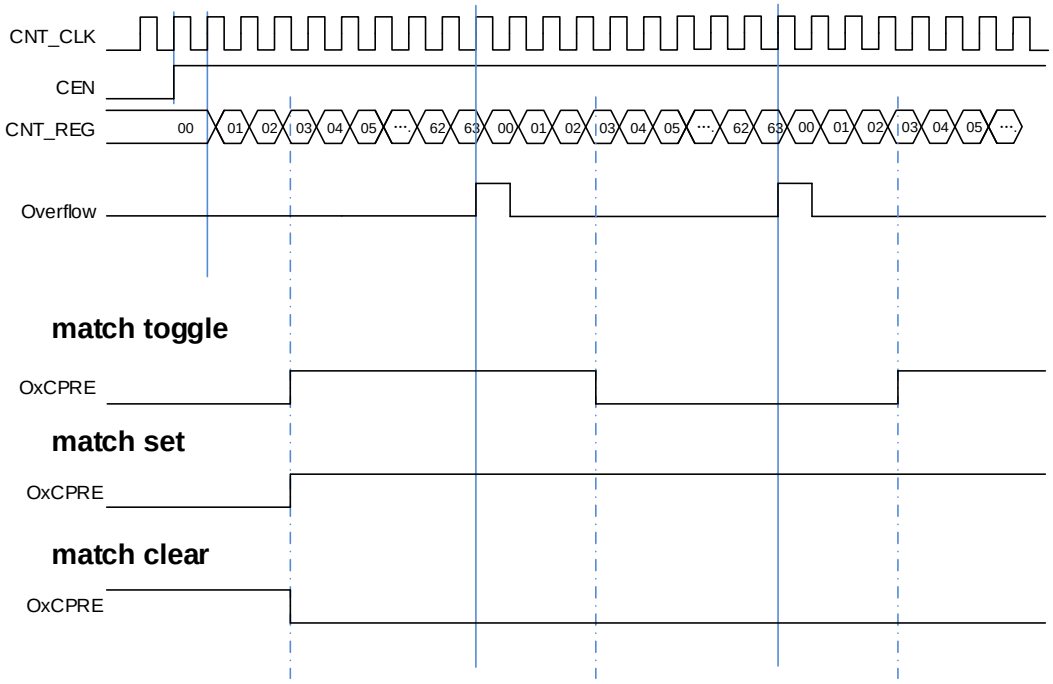
**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.
**Step2:** Compare mode configuration.

- Set the shadow enable mode by CHxCOMSEN/ MCHxCOMSEN.
- Set the output mode (set/clear/toggle) by CHxCOMCTL/ MCHxCOMCTL.
- Select the active polarity by CHxP/MCHxP/ MCHxFP.
- Enable the output by CHxEN/ MCHxEN.

**Step3:** Interrupt/DMA request enable configuration by CHxIE/ MCHxIE /CHxDEN/ MCHxDEN.
**Step4:** Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV/ TIMERx_MCHxCV.
The TIMERx_CHxCV/ TIMERx_MCHxCV can be changed onging to meet the expected waveform.
**Step5:** Start the counter by configuring CEN to 1.

*Figure 24-119. Output-compare in three modes* shows the three compare modes: toggle/set/clear. CARL=0x63, CHxVAL=0x3.

**Figure 24-119. Output-compare in three modes**



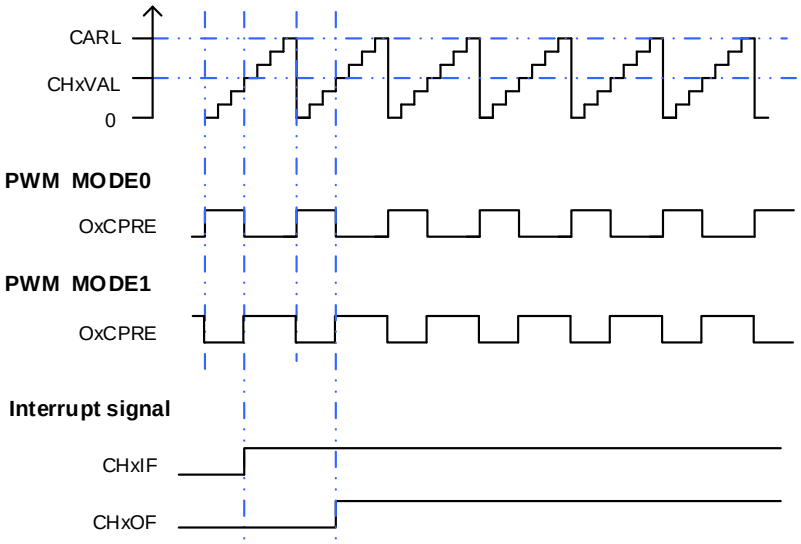

## PWM mode

In the PWM output mode (by setting the CHxCOMCTL/ MCHxCOMCTL bit to 4'b0110 (PWM mode 0) or to 4'b0111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV/ TIMERx_MCHxCV registers.

The EAPWM's period is determined by TIMERx_CAR and the duty cycle is determined by TIMERx_CHxCV/ TIMERx_MCHxCV. ***Figure 24-120. PWM mode timechart*** shows the EAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx_CHxCV/ TIMERx_MCHxCV is greater than the value of TIMERx_CAR, the output will be always active in PWM mode 0 (CHxCOMCTL/ MCHxCOMCTL =4'b0110). And if the value of TIMERx_CHxCV/ TIMERx_MCHxCV is greater than the value of TIMERx_CAR, the output will be always inactive in PWM mode 1 (CHxCOMCTL/ MCHxCOMCTL =4'b0111).

**Figure 24-120. PWM mode timechart**




## Channel output prepare signal

As is shown in ***Figure 24-117. Output compare logic (when MCHxMSEL = 2'00, x=0)*** and ***Figure 24-118. Output compare logic (when MCHxMSEL = 2'11, x=0)***, when TIMERx is configured in compare match output mode, a middle signal named OxCPRE or MOxCPRE (channel x output or multi mode channel x output prepare signal) will be generated before the channel outputs signal.

The OxCPRE and MOxCPRE signal have several types of output function. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit and the MOxCPRE signal type is defined by configuring the MCHxCOMCTL bit.

Take OxCPRE as an example for description below, these include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0/ PWM mode 1 output is another output type of OxCPRE which is setup by configuring the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value

and the TIMERx_CHxCV content. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/ 0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Outputs complementary

The outputs of CHx_O and MCHx_O have two situations:

- MCHxMSEL=2'b00: The MCHx_O output is independent from the CHx_O output;
- MCHxMSEL=2'b11: The outputs of MCHx_O and CHx_O are complementary and the MCHxCOMCTL bits are not used in the generation of the MCHx_O output.

Function of complementary is for a pair of channels, CHx_O and MCHx_O, the two output signals cannot be active at the same time. TIMERx's one pair of channel has this function. The complementary signals CHx_O and MCHx_O are controlled by a group of parameters: the CHxEN and MCHxEN bits in the TIMERx_CHCTL2 register, the POEN, ROS and IOS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register. The output polarity is determined by CHxP and MCHxP bits in the TIMERx_CHCTL2 register.

When the the outputs of CHx_O and MCHx_O are complementary, there are three situations: output enable、output off-state and output disabled. The details are shown in ***Table 24-18. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)***.

**Table 24-18. Complementary outputs controlled by parameters (MCHxMSEL =2'b11)**

<table>
<tr><th colspan="5">Complementary Parameters</th><th colspan="2">Output Status</th></tr>
<tr><th>POEN</th><th>ROS</th><th>IOS</th><th>CHxEN</th><th>MCHxEN</th><th>CHx_O</th><th>MCHx_O</th></tr>
<tr><td rowspan="5">0</td><td rowspan="5">0/1</td><td rowspan="3">0</td><td rowspan="2">0</td><td>0</td><td colspan="2">CHx_O / CHx_ON = LOW<br>CHx_O / CHx_ON output disable(1).</td></tr>
<tr><td>1</td><td colspan="2" rowspan="3">CHx_O/ CHx_ON output "off-state" (2):<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN. (3)</td></tr>
<tr><td rowspan="2">1</td><td>0</td></tr>
<tr><td>1</td></tr>
<tr><td>1</td><td>x</td><td>x</td><td colspan="2">CHx_O/ CHx_ON output "off-state":<br>the CHx_O/ CHx_ON output inactive level firstly: CHx_O = CHxP, CHx_ON = CHxNP; If the clock for deadtime generator is present, after a deadtime: CHx_O = ISOx, CHx_ON = ISOxN.</td></tr>
<tr><td>1</td><td>0</td><td>0/1</td><td>0</td><td>0</td><td colspan="2">CHx_O/MCHx_O = LOW<br>CHx_O/MCHx_O output disable.</td></tr>
</table>

| Complementary Parameters | | | | | Output Status | |
|---|---|---|---|---|---|---|
| POEN | ROS | IOS | CHxEN | MCHxEN | CHx_O | MCHx_O |
| | | | | 1 | CHx_O = LOW<br>CHx_O output disable. | MCHx_O=OxCPRE⊕<br>(4)MCHxP<br>MCHx_O output enable. |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | MCHx_O = LOW<br>MCHx_O output disable. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable. | MCHx_O = (! OxCPRE)(5)⊕<br>MCHxP.<br>MCHx_O output enable. |
| | 1 | | 0 | 0 | CHx_O = CHxP<br>CHx_O output “off-state”. | MCHx_O = MCHxP<br>MCHx_O output “off-state”. |
| | | | | 1 | CHx_O = CHxP<br>CHx_O output “off-state” | MCHx_O=OxCPRE⊕MCHxP<br>MCHx_O output enable |
| | | | 1 | 0 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | MCHx_O = MCHxP<br>MCHx_O output “off-state”. |
| | | | | 1 | CHx_O=OxCPRE⊕CHxP<br>CHx_O output enable | MCHx_O = (! OxCPRE)⊕<br>MCHxP<br>MCHx_O output enable. |

**Note:**

(1) output disable: the CHx_O / CHx_ON are disconnected to corresponding pins, the pin is floating with GPIO pull up/down setting which will be Hi-Z if no pull.

(2) “off-state”: CHx_O / CHx_ON output with inactive state (e.g., CHx_O = 0⊕CHxP = CHxP).

(3) See Break mode section for more details.

(4) ⊕: Xor calculate.

(5) (! OxCPRE)：the complementary output of the OxCPRE signal.

### Dead time insertion

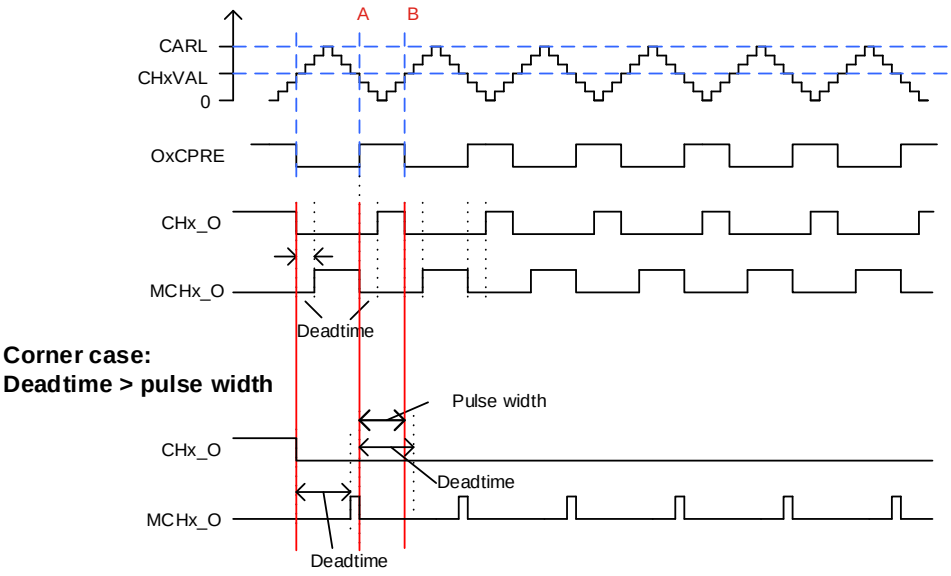
The dead time insertion is enabled when MCHxMSEL=2’b11 and both CHxEN and MCHxEN are configured to 1’b1, it is also necessary to configure POEN to 1. The field named DTCFG defines the dead time delay that can be used for all channels. Refer to the ***Complementary channel protection register (TIMERx_CCHP)*** for details about the delay time.

The dead time delay insertion ensures that two complementary signals are not active at the same time.

When the channel x match event (TIMERx_CNT = CHxVAL) occurs, OxCPRE will be toggled in PWM mode 0. At point A in ***Figure 24-121. Complementary output with dead-time insertion***, CHx_O signal remains at the low level until the end of the dead time delay, while MCHx_O signal will be cleared at once. Similarly, at point B when the channelx match event (TIMERx_CNT = CHxVAL) occurs again, OxCPRE is cleared, and CHx_O signal will be cleared at once, while MCHx_O signal remains at the low level until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example: the dead time delay is greater than or equal to the duty cycle of the CHx_O signal, then the CHx_O signal is always inactive (As shown in ***Figure 24-121. Complementary output with dead-time insertion***).

**Figure 24-121. Complementary output with dead-time insertion**




## Break function

The MCHx_O output is the inverse of the CHx_O output when the MCHxMSEL=2'b11 (and the MCHxCOMCTL bits are not used in the generation of the MCHx_O output). In this case, CHx_O and MCHx_O signals cannot be set to active level at the same time.

The general level3 timers have the BREAK0 function. The BREAK0 function can be enabled by setting the BRK0EN bit in the TIMERx_CCHP register. The break input polarity is configured by the BRK0P bit in TIMERx_CCHP register, the input is active on level.

In BREAK0 function, CHx_O and MCHx_O are controlled by the POEN, OAEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register.

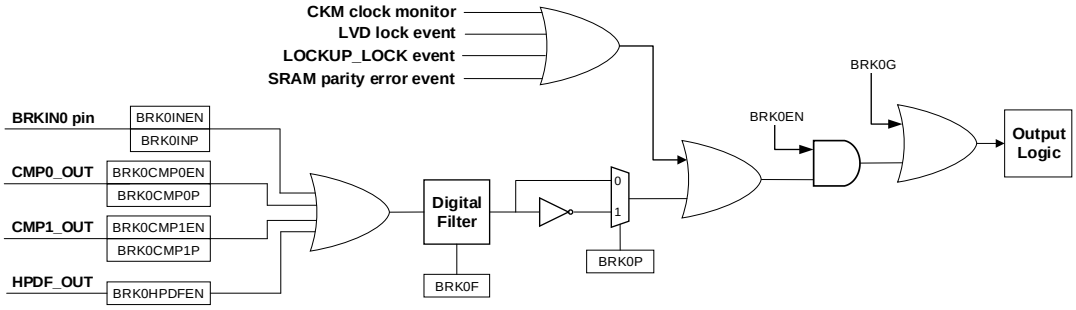
The break event is the result of logic ORed of all sources. The BREAK0 function can handle three types of event sources:

- External sources: coming from BRKIN0 input;
- System sources: HXTAL stuck event which is generated by Clock Monitor CKM in RCU, LVD lock event, Cortex®-M7 LOCKUP_LOCK event or SRAM parity error event;
- On-chip peripheral events: input by comparator output or HPDF watchdog output.

BREAK0 events can also be generated by software using BRK0G bit in the TIMERx_SWEVG register.

Refer to ***Figure 24-122. BREAK0 function logic diagram***, BRKIN0 can select GPIO pins from the TRIGSEL module, which can select by TRICSEL_TIMERxBTKIN registers.

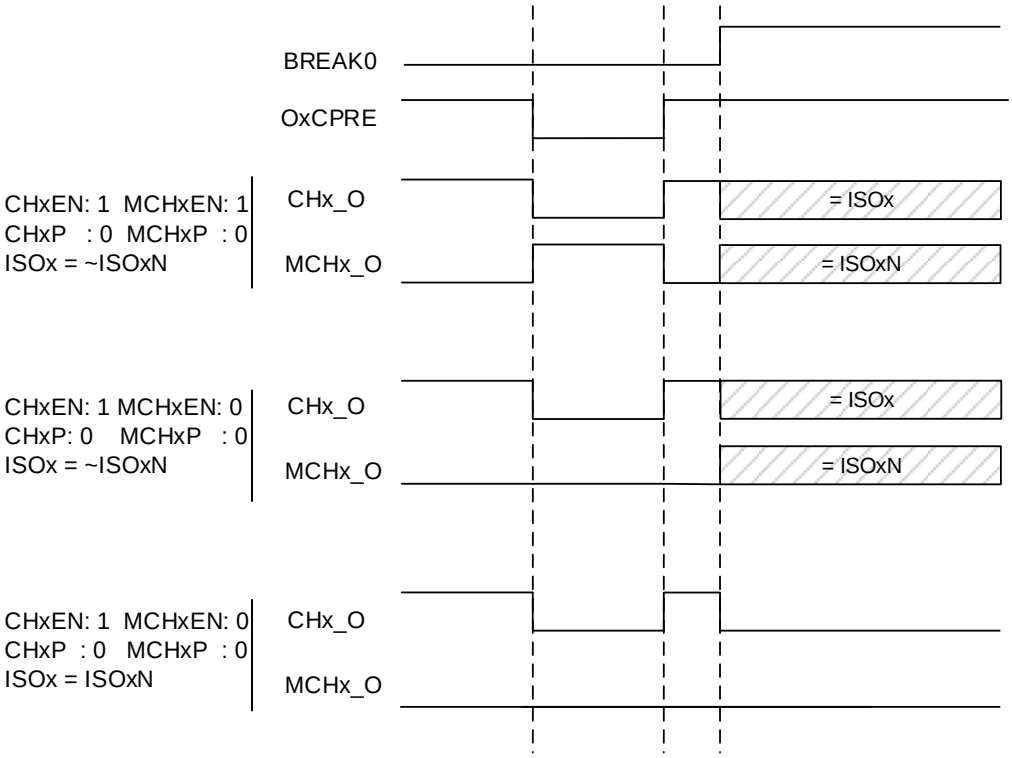
**Figure 24-122. BREAK0 function logic diagram**




When a BREAK0 event occurs, the outputs are force at an inactive level, or at a predefined level (either active or inactive) after a deadtime duration.

When the MCHxMSEL = 2'b11 and a break occurs, the POEN bit is cleared asynchronously. As soon as POEN is 0, the level of the CHx_O and MCHx_O outputs are determined by the ISOx and ISOxN bits in the TIMERx_CTL1 register. If IOS = 0, the timer releases the enable output, otherwise, the enable output remains high. When IOS=1, the output behavior of the channel is shown in ***Figure 24-123. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1)***. The complementary outputs are first in the reset state, and then the dead time generator is reactivated to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead time.

**Figure 24-123. Output behavior of the channel in response to BREAK0 (the break input high active and IOS=1)**




When a break occurs, the BRKIF bit in the TIMERx_INTF register will be set. If BRKIE is 1, an interrupt will be generated.

### Locked break function

The BRKIN0 input pin of general timer have the locked break function, this function can be enabled by setting the BRK0LK bit in the TIMERx_CCHP register.

When the locked break function is enabled, the BRKIN0 pins need to be configured to open-drain output mode with low level active (BRK0P=0 and BRK0IN0P=0). When any break source requests occur, the BRKIN0 pin can be forced to low level. If the break input polarity is active high (BRK0P=1 and BRK0IN0P=1), the locked break function is invalid.

When the break function is enabled (the BRK0EN =1), the BRKIN0 pin can be forced to low level with the BRK0G bit setting to 1 by software.

When the break function is disabled (the BRK0EN =0), setting the BRK0G bit will have no effect on the BRKIN0 pin. The BRK0F bit will set and the channel outputs will be in a safe state.

The BRKIN0 pin can be released by setting the BRK0REL bit in the TIMERx_CCHP register. When the break input sources are inactive, the BRK0REL bit will cleared by hardware and the BRKIN0 pin will restore the locked break function.

In the following two cases, the BRKIN0 pin cannot be released:

- Break input sources are active: the BRK0REL bit is set to 1 and the BRKIN0 pin locked break function is released. The break events are still active, because the break input sources are still active.
- POEN=1: when the channel outputs are enabled, the BRKIN0 pin cannot be released even if the BRK0REL is set.

**Table 24-19. Break function input pins locked/ released conditions**

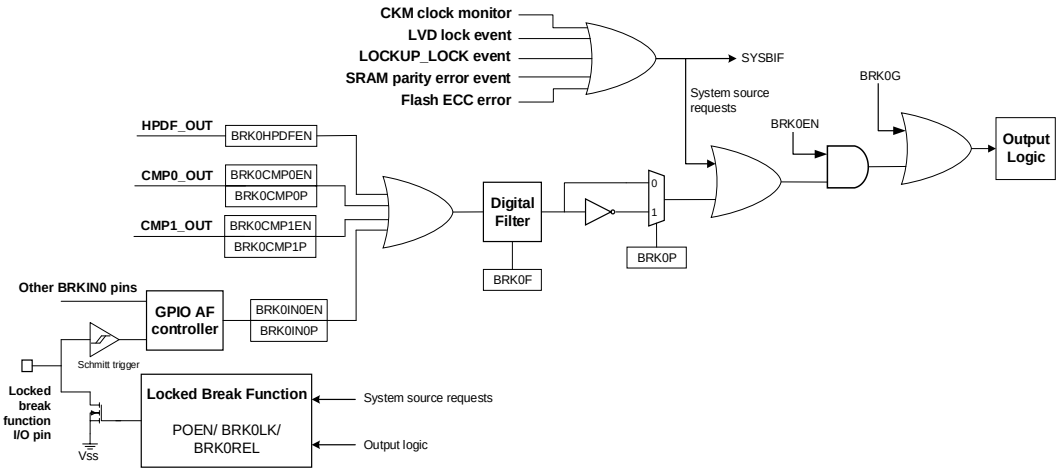
| POEN | BRK0LK | BRK0REL | Break input pin state |
|---|---|---|---|
| 0 | 1 | 0 | Locked |
| | 1 | 1 | Released |

The locked break function of the BREAK0 input pin BRKIN0 is enabled by default (BRK0REL=0). When the BREAK0 events occur, the following steps can be used to reconfigure the locked break function:

- Set the BRK0REL bit to 1 and released the BRKIN0 pin;
- The software waits for the system break sources inactive, and then clears the SYSBIF flag;
- The software polls the state of BRK0REL bit, until the BRK0REL bit is cleared (cleared by hardware).

Then the locked break function of BREAK0 input pin is re-enabled, and the channel outputs can be restored by setting the POEN bit to 1 by software.

**Figure 24-124. BRKIN0 pin logic with BREAK0 function**



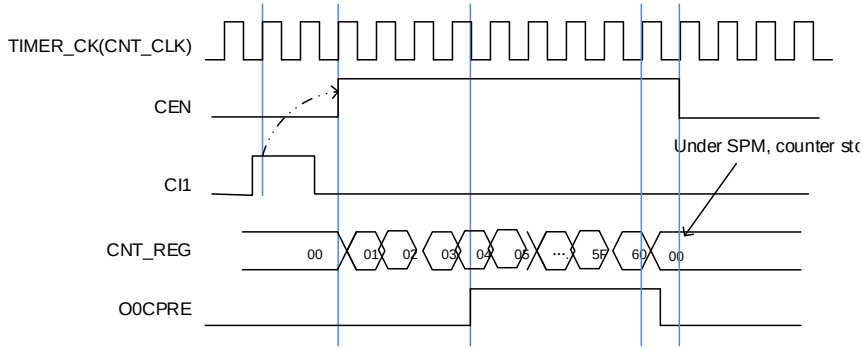

## Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. If SPM is set, the counter will be cleared and stopped automatically when the next update event occurs. In order to get a pulse waveform, the TIMERx is configured to PWM mode or compare mode by CHxCOMCTL or MCHxCOMCTL bits.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. Setting the CEN bit to 1 or a trigger signal edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 by software, the counter will be stopped and its value will be held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the active edge of trigger which sets the CEN bit to 1 will enable the counter. However, there exists several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE/ MOxCPRE signals will change to, as the compare match event occurs without taking the comparison result into account.

**Figure 24-125. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60**



### Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx_DMACFG and TIMERx_DMATB. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx_DMATB, then DMA will access the TIMERx_DMATB. In fact, register TIMERx_DMATB is only a buffer; timer will map the TIMERx_DMATB to an internal register, appointed by the field of DMATA in TIMERx_DMACFG. If the field of DMATC in TIMERx_DMACFG is 0(1 transfer), then the timer's DMA request is finished. While if TIMERx_DMATC is not 0, such as 3(4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8, DMATA+0xc at the next 3 accesses to TIMERx_DMATB. In a word, one-time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### UPIF bit backup

The UPIF bit backup function is enabled by setting UPIFBUEN in the TIMERx_CTL0 register. The UPIF and UPIFBU bits are fully synchronized and without latency.

By using this function, the UPIF bit in the TIMERx_INTF register will be backuped to the UPIFBU bit in the TIMERx_CNT register. This can avoid conflicts when reading the counter and interrupt processing.

### Timer debug mode

When the Cortex®-M7 halted, and the TIMERx_HOLD configuration bit in DBG_CTL1 register set to 1, the TIMERx counter stops.

### 24.4.5. Register definition (TIMERx, x=15,16)

TIMER15 base address: 0x4001 4400

TIMER16 base address: 0x4001 4800

#### Control register 0 (TIMERx_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | UPIFBUEN | Reserved | CKDIV[1:0] | | ARSE | Reserved | | | SPM | UPS | UPDIS | CEN |
| | | | | rw | | rw | | rw | | | | rw | rw | rw | rw |

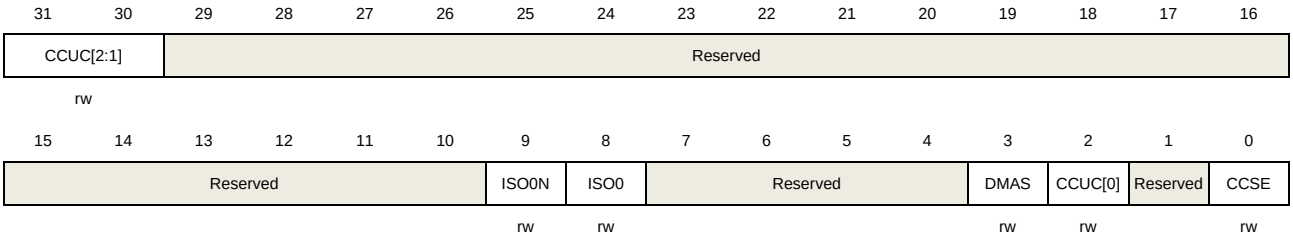
| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | UPIFBUEN | UPIF bit backup enable<br>0: Backup disable. UPIF bit is not backuped to UPIFBU bit in TIMERx_CNT register.<br>1: Backup enabled. UPIF bit is backuped to UPIFBU bit in TIMERx_CNT register. |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | CKDIV[1:0] | Clock division<br>The CKDIV bits can be configured by software to specify division ratio between CK_TIMER (the timer clock) and DTS (the dead time and sampling clock) which is used for the dead time generator and the digital filter.<br>00: $f_{DTS}$= $f_{CK_TIMER}$<br>01: $f_{DTS}$= $f_{CK_TIMER}$ /2<br>10: $f_{DTS}$= $f_{CK_TIMER}$ /4<br>11: Reserved |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR register is disabled<br>1: The shadow register for TIMERx_CAR register is enabled |
| 6:4 | Reserved | Must be kept at reset value. |
| 3 | SPM | Single pulse mode<br>0: Single pulse mode is disabled. Counter continues after an update event.<br>1: Single pulse mode is enabled. The CEN bit is cleared by hardware and the counter stops at next update event. |

| | | |
|---|---|---|
| 2 | UPS | Update source<br>This bit is used to select the update event sources by software.<br>0: Any of the following events generates an update interrupt or a DMA request:<br>– The UPG bit is set.<br>– The counter generates an overflow event.<br>– The slave mode controller generates an update event.<br>1: Only counter overflow generates an update interrupt or a DMA request. |
| 1 | UPDIS | Update disable<br>This bit is used to enable or disable the update event generation.<br>0: Update event enable. The update event is generated and the buffered registers are loaded with their preloaded values when one of the following events occurs:<br>– The UPG bit is set.<br>– The counter generates an overflow event.<br>– The slave mode controller generates an update event.<br>1: Update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or the slave mode controller generates a hardware reset event. |
| 0 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable<br>The CEN bit must be set by software when timer works in external clock mode, pause mode or decoder mode. While in event mode, the hardware can set the CEN bit automatically. |

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | CCUC[2:1] | Commutation control shadow register update control<br>Refer to CCUC [0] description. |
| 29:10 | Reserved | Must be kept at reset value. |
| 9 | ISO0N | Idle state of multi mode channel 0 complementary output |

| | | |
|---|---|---|
| | | 0: When POEN bit is reset, MCH0_O is set low.<br>1: When POEN bit is reset, MCH0_O is set high.<br>This bit can be modified only when PROT[1:0] bits in TIMERx_CCHP register is 00. |
| 8 | ISO0 | Idle state of channel 0 output<br>0: When POEN bit is reset, CH0_O is set low.<br>1: When POEN bit is reset, CH0_O is set high.<br>The CH0_O output changes after a dead time if MCH0_O is implemented. This bit can be modified only when PROT[1:0] bits in TIMERx_CCHP register is 00. |
| 7:4 | Reserved | Must be kept at reset value. |
| 3 | DMAS | DMA request source selection<br>0: DMA request of CHx/MCHx is sent when capture/compare event occurs.<br>1: DMA request of channel CHx/MCHx is sent when update event occurs. |
| 2 | CCUC[0] | Commutation control shadow register update control<br>The CCUC[2:1] and CCUC[0] field are used to control the commutation control shadow register update. When the commutation control shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are enabled (CCSE=1), the update control of the shadow registers with the CCUC[2:0] bit-field are shown as below:<br>000: The shadow registers update when CMTG bit is set.<br>001: Reserved.<br>100: The shadow registers update when the counter generates an overflow event.<br>Others: Reserved<br>When a channel does not have a complementary output, this bit has no effect.<br>**Note**: When CCUC[2:0] bit-field are set to 100, the update of the shadow registers also considers the value the CCUSEL bit in the TIMERx_CFG register. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | CCSE | Commutation control shadow enable<br>0: The shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are disabled.<br>1: The shadow registers (for CHxEN, MCHxEN and CHxCOMCTL bits) are enabled. After these bits have been written, they are updated when commutation event comes.<br>When a channel does not have a complementary output, this bit has no effect. |

### DMA and interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | MCH0DEN | Reserved | | | MCH0IE | Reserved | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | rw | | | | rw | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | CH0DEN | UPDEN | BRKIE | Reserved | CMTIE | Reserved | | | CH0IE | UPIE |
| | | | | | | rw | rw | rw | | rw | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | MCH0DEN | Multi mode channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH0SEL[1:0] = 2b'00). |
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | MCH0IE | Multi mode channel 0 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled<br>**Note:** This bit just used for channel input and output independent mode (when MMCH0SEL[1:0] = 2b'00). |
| 19:10 | Reserved | Must be kept at reset value. |
| 9 | CH0DEN | Channel 0 capture/compare DMA request enable<br>0: Disabled<br>1: Enabled |
| 8 | UPDEN | Update DMA request enable<br>0: Disabled<br>1: Enabled |
| 7 | BRKIE | Break interrupt enable<br>0: Disabled<br>1: Enabled |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CMTIE | Commutation interrupt enable<br>0: Disabled<br>1: Enabled |
| 4:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IE | Channel 0 capture/compare interrupt enable<br>0: Disabled<br>1: Enabled |
| 0 | UPIE | Update interrupt enable<br>0: Disabled |

1: Enabled

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | MCH0OF | Reserved | | | MCH0IF | Reserved | | | |
| | | | | | | | rc_w0 | | | | rc_w0 | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CH0OF | Reserved | BRK0IF | Reserved | CMTIF | Reserved | | | CH0IF | UPIF |
| | | | | | | rc_w0 | | rc_w0 | | rc_w0 | | | | rc_w0 | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | MCH0OF | Multi mode channel 0 over capture flag<br>When multi mode channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while MCH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | MCH0IF | Multi mode channel 0 capture/compare interrupt flag<br>This flag is set by hardware and cleared by software.<br>If multi mode channel 0 is in input mode, this flag is set when a capture event occurs.<br>If multi mode channel 0 is in output mode, this flag is set when a compare event occurs.<br>If multi mode channel 0 is set to input mode, this bit will be reset by reading TIMERx_MCH0CV.<br>0: No multi mode channel 0 capture/compare interrupt occurred<br>1: Multi mode channel 0 capture/compare interrupt occurred |
| 19:10 | Reserved | Must be kept at reset value. |
| 9 | CH0OF | Channel 0 over capture flag<br>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.<br>0: No over capture interrupt occurred<br>1: Over capture interrupt occurred |
| 8 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 7 | BRK0IF | BREAK0 interrupt flag<br>This flag is set by hardware when the BREAK0 input is active, and cleared by software if the BREAK0 input is not at active level.<br>0: No active level on BREAK0 input has been detected.<br>1: An active level on BREAK0 input has been detected. |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CMTIF | Channel commutation interrupt flag<br>This flag is set by hardware when the commutation event of channel occurs, and cleared by software.<br>0: No channel commutation interrupt occurred<br>1: Channel commutation interrupt occurred |
| 4:2 | Reserved | Must be kept at reset value. |
| 1 | CH0IF | Channel 0 capture/compare interrupt flag<br>This flag is set by hardware and cleared by software.<br>If channel 0 is in input mode, this flag is set when a capture event occurs. If channel 0 is in output mode, this flag is set when a compare event occurs.<br>If channel 0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV.<br>0: No channel 0 interrupt occurred<br>1: Channel 0 interrupt occurred |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware when an update event occurs and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | MCH0G | Reserved | | | |
| | | | | | | | | | | | w | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | BRK0G | Reserved | CMTG | Reserved | | | CH0G | UPG |
| | | | | | | | | w | | w | | | | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | MCH0G | Multi mode channel 0 capture or compare event generation.<br>This bit is set by software to generate a capture or compare event in multi mode |

| | | |
|---|---|---|
| | | channel 0, it is automatically cleared by hardware. When this bit is set, the MCH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if multi mode channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_MCH0CV register, and the MCH0OF flag is set if the MCH0IF flag has been set.<br>0: No generate a multi mode channel 0 capture or compare event<br>1: Generate a multi mode channel 0 capture or compare event |
| 19:8 | Reserved | Must be kept at reset value. |
| 7 | BRK0G | BREAK0 event generation<br>This bit is set by software to generate an event and cleared by hardware automatically. When this bit is set, the POEN bit will be cleared and BRK0IF flag will be set, related interrupt can occur if enabled.<br>0: No generate a break event<br>1: Generate a break event |
| 6 | Reserved | Must be kept at reset value. |
| 5 | CMTG | Channel commutation event generation<br>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, MCHxEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).<br>0: No affect<br>1: Generate channel commutation update event |
| 4:2 | Reserved | Must be kept at reset value. |
| 1 | CH0G | Channel 0 capture or compare event generation<br>This bit is set by software to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag will be set, and the corresponding interrupt or DMA request will be sent if enabled. In addition, if channel 0 is configured in input mode, the current value of the counter is captured to TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag has been set.<br>0: No generate a channel 0 capture or compare event<br>1: Generate a channel 0 capture or compare event |
| 0 | UPG | Update event generation<br>This bit can be set by software, and automatically cleared by hardware. When this bit is set, the counter is cleared if the up counting mode is selected. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Channel control register 0 (TIMERx_CHCTL0)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Refer to CH0MS[1:0] description. |
| 29:17 | Reserved | Must be kept at reset value. |
| 16 | CH0COMCTL[3] | Channel 0 compare output control<br>Refer to CH0COMCTL[2:0] description |
| 15:7 | Reserved | Must be kept at reset value. |
| 6:4 | CH0COMCTL[2:0] | Channel 0 compare output control<br>The CH0COMCTL[3] and CH0COMCTL[2:0] bit-field control the behavior of O0CPRE which drives CH0_O. The active level of O0CPRE is high, while the active level of CH0_O depends on CH0P bit.<br>**Note**: When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, This bit-field controls the behavior of O0CPRE which drives CH0_O and MCH0_O. The active level of O0CPRE is high, while the active level of CH0_O and MCH0_O depends on CH0P and MCH0P bits.<br>0000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.<br>0010: Clear the channel output on match. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.<br>0011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.<br>0100: Force low. O0CPRE is forced low level. |

| | | |
|---|---|---|
| | | 0101: Force high. O0CPRE is forced high level.<br>0110: PWM mode 0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV, otherwise it is inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV, otherwise it is active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV, otherwise it is inactive.<br>1000~1111: Reserved.<br>**Note**:<br>If configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “Timing” mode to “PWM” mode or the result of the comparison changes.<br>When the outputs of CH0 and MCH0 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000 (compare mode). |
| 3 | CH0COMSEN | Channel 0 compare output shadow enable<br>When this bit is set, the shadow register of TIMERx_CH0CV register which updates at each update event will be enabled.<br>0: Channel 0 output compare shadow disabled<br>1: Channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. The CH0MS[2:0] bit-field is writable only when the channel is not active (When MCH0MSEL[1:0] = 2b’00, the CH1EN bit in TIMERx_CHCTL2 register is reset; when MCH0MSEL[1:0] = 2b’11, the CH0EN and MCH0EN bits in TIMERx_CHCTL2 register are reset).<br>000: Channel 0 is configured as output.<br>001: Channel 0 is configured as input, IS0 is connected to CI0FE0.<br>010: Reserved<br>011: Reserved<br>100: Channel 0 is configured as input, IS0 is connected to MCI0FE0.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | CH0MS[2] | Channel 0 I/O mode selection<br>Refer to CH0MS[1:0] description. |
| 29:8 | Reserved | Must be kept at reset value. |
| 7:4 | CH0CAPFLT[3:0] | Channel 0 input capture filter control<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6.<br>0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | CH0CAPPSC[1:0] | Channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | CH0MS[1:0] | Channel 0 I/O mode selection<br>Same as output compare mode. |

### Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | MCH0P | MCH0EN | CH0P | CH0EN |
| | | | | | | | | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | MCH0P | Multi mode channel 0 output polarity<br>When Multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, this bit specifies the MCH0_O output signal polarity.<br>0: Multi mode channel 0 output active high<br>1: Multi mode channel 0 output active low<br>When CH0 is configured in input mode, in conjunction with CH0P, this bit is used to define the polarity of CH0.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 2 | MCH0EN | Multi mode channel 0 capture/compare enable<br>When multi mode channel 0 is configured in output mode, setting this bit enables MCH0_O signal in active state. When multi mode channel 0 is configured in input mode, setting this bit enables the capture event in multi mode channel 0.<br>0: Multi mode channel 0 disabled<br>1: Multi mode channel 0 enabled |
| 1 | CH0P | Channel 0 capture/compare polarity<br>When channel 0 is configured in output mode, this bit specifies the output signal polarity.<br>0: Channel 0 active high<br>1: Channel 0 active low<br>When channel 0 is configured in input mode, these bits specifie the channel 0 input signal's polarity. [MCH0P, CH0P] will select the active trigger or capture polarity for channel 0 input signals.<br>00: channel 0 input signal's rising edge is the active signal for capture or trigger operation in slave mode. And channel 0 input signal will not be inverted.<br>01: channel 0 input signal's falling edge is the active signal for capture or trigger operation in slave mode. And channel 0 input signal will be inverted.<br>10: Reserved.<br>11: Noninverted/both channel 0 input signal's edges.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10. |
| 0 | CH0EN | Channel 0 capture/compare enable<br>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables |

the capture event in channel 0.

0: Channel 0 disabled

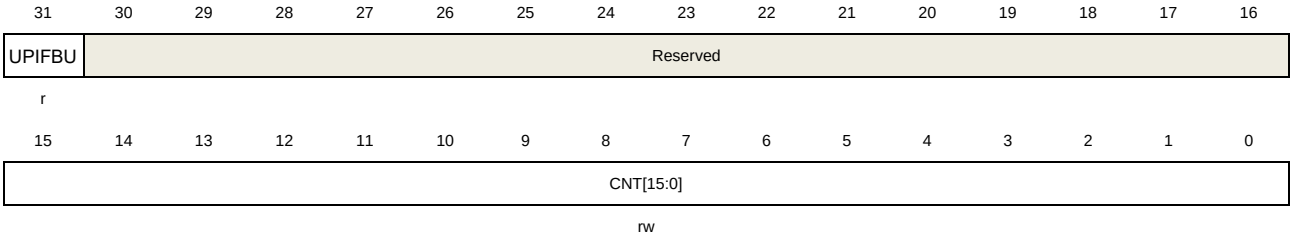
1: Channel 0 enabled

### Counter register (TIMERx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



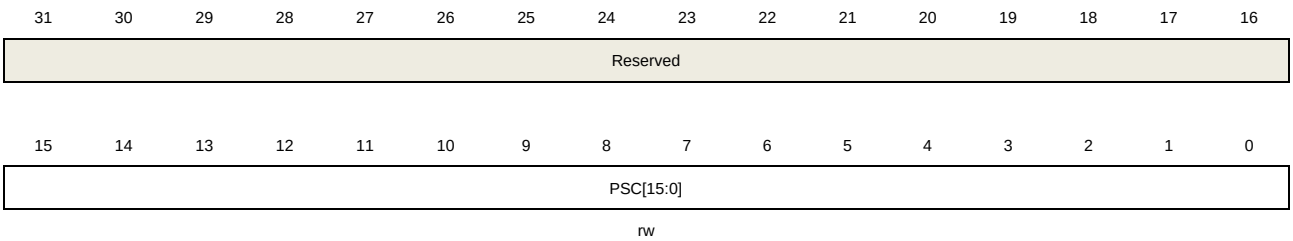

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register, and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:0 | CNT[15:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



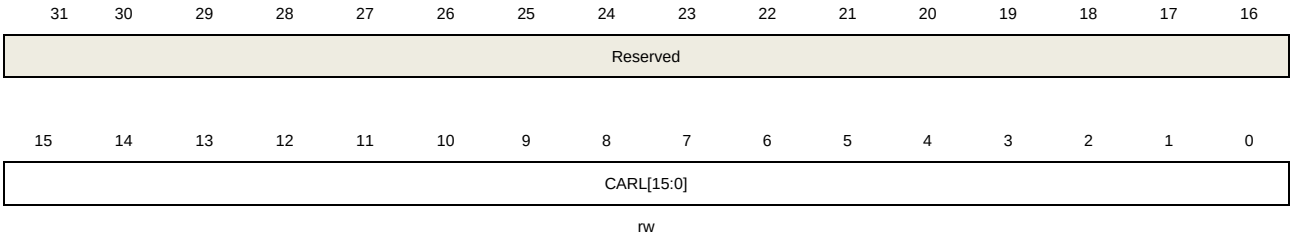

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of |

this bit-field will be loaded to the corresponding shadow register at every update event.

### Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C
Reset value: 0x0000 FFFF

This register has to be accessed by word (32-bit).

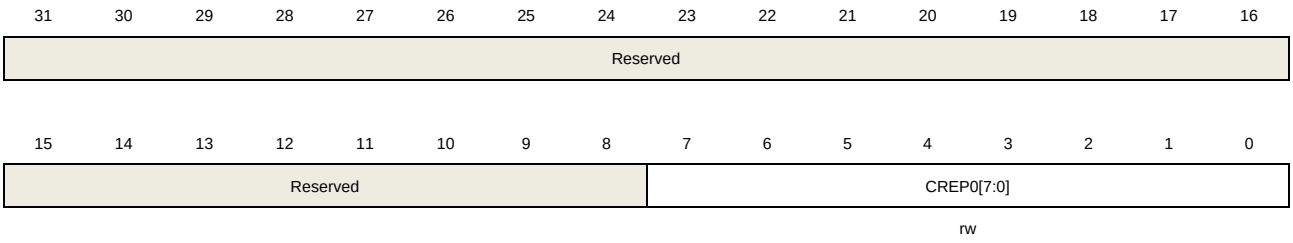



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CARL[15:0] | Counter auto reload value<br>This bit-field specifies the auto reload value of the counter. |

### Counter repetition register 0 (TIMERx_CREP0)

Address offset: 0x30
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | CREP0[7:0] | Counter repetition value 0<br>This bit-field specifies the update event generation rate. Each time the repetition counter counts down to zero, an update event will be generated. The update rate of the shadow registers is also affected by this bit-field when these shadow registers are enabled.<br>**Note**: This bit-field just used with CREPSEL =0 (in TIMERx_CFG register). |

### Channel 0 capture/compare value register (TIMERx_CH0CV)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CH0VAL[15:0] | Capture/compare value of channel 0<br>When channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Complementary channel protection register (TIMERx_CCHP)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | BRK0LK | Reserved | BRK0REL | Reserved | | | | | | BRK0F[3:0] | | | |
| | rw | | rw | | rw | | | | | | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| POEN | OAEN | BRK0P | BRK0EN | ROS | IOS | PROT[1:0] | | DTCFG[7:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | BRK0LK | BREAK0 input locked<br>0: BREAK0 input in input mode<br>1: BREAK0 input in locked mode<br>When the BRK0LK is set to 1, the BREAK0 input is configured in open drain output mode.<br>Any active break event asserts a low logic level on the BREAK0 input to indicate an internal break event to external devices. |

This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

**Note**: Every write operation to this bit needs a delay of 1 APB clock to active.

27 Reserved Must be kept at reset value.

26 BRK0REL BREAK0 input released

This bit is cleared by hardware when the break input is invalid.

0: BREAK0 input is unreleased

1: BREAK0 input is released

The locked output control (open drain mode in Hi-z state) is released by setting this bit with software. And when the fault is disappeared, this bit will reset by hardware.

**Note**: Every write operation to this bit needs a delay of 1 APB clock to active.

25:20 Reserved Must be kept at reset value.

19:16 BRK0F[3:0] BREAK0 input signal filter

An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample BREAK0 input signal and the length of the digital filter applied to BREAK0.

0000: Filter disabled. BREAK0 act asynchronously, N=1

0001: $f_{SAMP} = f_{CK_TIMER}$, N=2

0010: $f_{SAMP} = f_{CK_TIMER}$, N=4

0011: $f_{SAMP} = f_{CK_TIMER}$, N=8

0100: $f_{SAMP} = f_{DTS}/2$, N=6

0101: $f_{SAMP} = f_{DTS}/2$, N=8

0110: $f_{SAMP} = f_{DTS}/4$, N=6

0111: $f_{SAMP} = f_{DTS}/4$, N=8

1000: $f_{SAMP} = f_{DTS}/8$, N=6

1001: $f_{SAMP} = f_{DTS}/8$, N=8

1010: $f_{SAMP} = f_{DTS}/16$, N=5

1011: $f_{SAMP} = f_{DTS}/16$, N=6

1100: $f_{SAMP} = f_{DTS}/16$, N=8

1101: $f_{SAMP} = f_{DTS}/32$, N=5

1110: $f_{SAMP} = f_{DTS}/32$, N=6

1111: $f_{SAMP} = f_{DTS}/32$, N=8

This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00.

15 POEN Primary output enable

This bit is set by software or automatically set by hardware depending on the OAEN bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and MCHx_O) if the corresponding enable bits (CHxEN, MCHxEN in TIMERx_CHCTL2 register) have been set.

0: Channel outputs are disabled or forced to idle state.

| | | |
|---|---|---|
| | | 1: Channel outputs are enabled. |
| 14 | OAEN | Output automatic enable<br>This bit specifies whether the POEN bit can be set automatically by hardware.<br>0: POEN cannot be set by hardware.<br>1: POEN can be set by hardware automatically at the next update event, if the break input is not active.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 13 | BRK0P | BREAK0 input signal polarity<br>This bit specifies the polarity of the BREAK0 input signal.<br>0: BREAK0 input active low<br>1: BREAK0 input active high<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 12 | BRK0EN | BREAK0 input signal enable<br>This bit can be set to enable the BREAK0 input signal<br>0: BREAK0 input disabled<br>1: BREAK0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 11 | ROS | Run mode “off-state” enable<br>When POEN bit is set (Run mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2’b11)***.<br>0: “off-state” disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.<br>1: “off-state” enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11. |
| 10 | IOS | Idle mode “off-state” enable<br>When POEN bit is reset (Idle mode), this bit can be set to enable the “off-state” for the channels which has been configured in output mode. Please refer to ***Table 24-15. Complementary outputs controlled by parameters (MCHxMSEL =2’b11)***.<br>0: “off-state” disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.<br>1: “off-state” enabled. No matter the CHxEN/CHxNEN bits, the channels are “off-state”.<br>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is |

10 or 11.

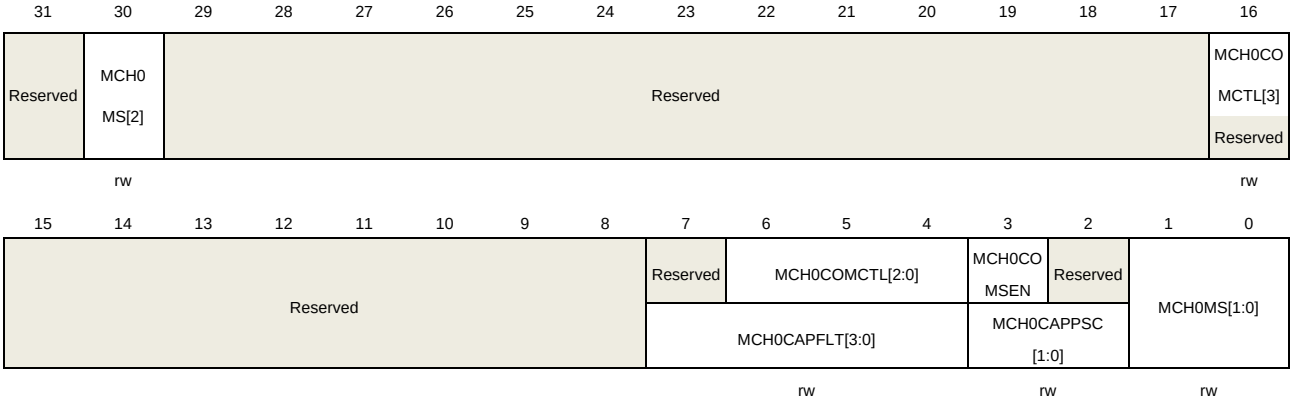
| | | |
|---|---|---|
| 9:8 | PROT[1:0] | Complementary register protect control<br>This bit-field specifies the write protection property of registers.<br>00: Protect disabled. No write protection.<br>01: PROT mode 0. The ISOx/ISOxN bits in TIMERx_CTL1 register, the BRK0EN/ BRK0P/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.<br>10: PROT mode 1. In addition to the registers in PROT mode 0, the CHxP/MCHxP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode), the ROS/IOS bits in TIMERx_CCHP register are writing protected.<br>11: PROT mode 2. In addition to the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN/ CHxCOMADDSEN/ MCHxCOMCTL/ MCHxCOMSEN bits in TIMERx_CHCTL0 and TIMERx_MCHCTL0 registers (if the related channel is configured in output) are writing protected.<br>This bit-field can be written only once after the system reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected. |
| 7:0 | DTCFG[7:0] | Dead time configuration<br>This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between the value of DTCFG and the duration of dead-time is as follow:<br>DTCFG[7:5] = 3'b0xx: DT value = DTCFG[7:0] * $t_{DT}$, $t_{DT}$ = $t_{DTS}$.<br>DTCFG[7:5] = 3'b10x: DT value = (64+DTCFG[5:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*2.<br>DTCFG[7:5] = 3'b110: DT value = (32+DTCFG[4:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*8.<br>DTCFG[7:5] = 3'b111: DT value = (32+DTCFG[4:0]) * $t_{DT}$, $t_{DT}$ = $t_{DTS}$*16.<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### Multi mode channel control register 0 (TIMERx_MCHCTL0)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




**Output compare mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | MCH0MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH0MS[1:0] description. |
| 29:17 | Reserved | Must be kept at reset value. |
| 16 | MCH0COMCTL[3] | Multi mode channel 0 compare output control.<br>Refer to MCH0COMCTL[2:0] description. |
| 15:7 | Reserved | Must be kept at reset value. |
| 6:4 | MCH0COMCTL[2:0] | Multi mode channel 0 output compare control<br>When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'00, the MCH0COMCTL[3] and MCH0COMCTL[2:0] bit-field control the behavior of MO0CPRE which drives MCH0_O. The active level of MO0CPRE is high, while the active level of MCH0_O depends on MCH0FP[1:0] bits.<br>**Note**: When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'11, the CH0COMCTL[2:0] bit-field controls the behavior of O0CPRE which drives CH0_O and MCH0_O, while the active level of CH0_O and MCH0_O depends on CH0P and MCH0P bits.<br>0000: Timing mode. The MO0CPRE signal keeps stable, independent of the comparison between the register TIMERx_MCH0CV and the counter TIMERx_CNT.<br>0001: Set the channel output on match. MO0CPRE signal is forced high when the counter matches the output compare register TIMERx_MCH0CV.<br>0010: Clear the channel output on match. MO0CPRE signal is forced low when the counter matches the output compare register TIMERx_MCH0CV.<br>0011: Toggle on match. MO0CPRE toggles when the counter matches the output compare register TIMERx_MCH0CV.<br>0100: Force low. MO0CPRE is forced low level.<br>0101: Force high. MO0CPRE is forced high level.<br>0110: PWM mode 0. When counting up, MO0CPRE is active as long as the counter is smaller than TIMERx_MCH0CV, otherwise it is inactive. When counting down, MO0CPRE is inactive as long as the counter is larger than TIMERx_MCH0CV, otherwise it is active.<br>0111: PWM mode 1. When counting up, MO0CPRE is inactive as long as the counter is smaller than TIMERx_MCH0CV, otherwise it is active. When counting down, MO0CPRE is active as long as the counter is larger than TIMERx_MCH0CV, otherwise it is inactive.<br>1000~1111: Reserved.<br>If configured in PWM mode, the MO0CPRE level changes only when the output compare mode switches from "Timing" mode to "PWM" mode or the result of the comparison changes.<br>When the outputs of CH0 and MCH0 are complementary, this bit-field is preloaded. If CCSE =1, this bit-field will only be updated when a channel commutation event is |

| | | |
|---|---|---|
| | | generated.<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0NMS bit-field is 00(compare mode). |
| 3 | MCH0COMSEN | Multi mode channel 0 output compare shadow enable<br>When this bit is set, the shadow register of TIMERx_MCH0CV register which updates at each update event will be enabled.<br>0: Multi mode channel 0 output compare shadow disabled<br>1: Multi mode channel 0 output compare shadow enabled<br>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).<br>This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and MCH0MS bit-field is 00. |
| 2 | Reserved | Must be kept at reset value. |
| 1:0 | MCH0MS[1:0] | Multi mode channel 0 I/O mode selection<br>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active (MCH0EN bit in TIMERx_CHCTL2 register is reset).<br>000: Multi mode channel 0 is configured as output.<br>001: Multi mode channel 0 is configured as input, MIS0 is connected to MCI0FEM0.<br>010: Reserved.<br>011: Reserved.<br>100: Multi mode channel 0 is configured as input, MIS0 is connected to CI0FEM0.<br>101~111: Reserved. |

**Input capture mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | MCH0MS[2] | Multi mode channel 0 I/O mode selection<br>Refer to MCH0MS[1:0] description. |
| 29:8 | Reserved | Must be kept at reset value. |
| 7:4 | MCH0CAPFLT[3:0] | Multi mode channel 0 input capture filter control.<br>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample MCI0 input signal and the length of the digital filter applied to MCI0.<br>0000: Filter disabled, $f_{SAMP}=f_{DTS}$, N=1.<br>0001: $f_{SAMP}=f_{CK_TIMER}$, N=2.<br>0010: $f_{SAMP}= f_{CK_TIMER}$, N=4.<br>0011: $f_{SAMP}= f_{CK_TIMER}$, N=8.<br>0100: $f_{SAMP}=f_{DTS}/2$, N=6.<br>0101: $f_{SAMP}=f_{DTS}/2$, N=8.<br>0110: $f_{SAMP}=f_{DTS}/4$, N=6. |

| | | |
|---|---|---|
| | | 0111: $f_{SAMP}=f_{DTS}/4$, N=8.<br>1000: $f_{SAMP}=f_{DTS}/8$, N=6.<br>1001: $f_{SAMP}=f_{DTS}/8$, N=8.<br>1010: $f_{SAMP}=f_{DTS}/16$, N=5.<br>1011: $f_{SAMP}=f_{DTS}/16$, N=6.<br>1100: $f_{SAMP}=f_{DTS}/16$, N=8.<br>1101: $f_{SAMP}=f_{DTS}/32$, N=5.<br>1110: $f_{SAMP}=f_{DTS}/32$, N=6.<br>1111: $f_{SAMP}=f_{DTS}/32$, N=8. |
| 3:2 | MCH0CAPPSC[1:0] | Multi mode channel 0 input capture prescaler<br>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when MCH0EN bit in TIMERx_CHCTL2 register is cleared.<br>00: Prescaler disabled, capture is done on each channel input edge.<br>01: Capture is done every 2 channel input edges.<br>10: Capture is done every 4 channel input edges.<br>11: Capture is done every 8 channel input edges. |
| 1:0 | MCH0MS[1:0] | Multi mode channel 0 I/O mode selection<br>Same as output compare mode |

### Multi mode channel control register 2 (TIMERx_MCHCTL2)

Address offset: 0x50

Reset value: 0x0000 0000

This register can be accessed by half-word(16-bit) or word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | MCH0FP[1:0] | |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1:0 | MCH0FP[1:0] | Multi mode channel 0 capture/compare free polarity<br>When multi mode channel 0 is configured in output mode, and the MCH0MSEL[1:0] = 2b'00, these bits specifie the multi mode channel 0 output signal polarity.<br>00: Multi mode channel 0 active high<br>01: Multi mode channel 0 active low<br>10: Reserved.<br>11: Reserved.<br>When multi mode channel 0 is configured in input mode, these bits specifie the multi mode channel 0 input signal's polarity. MCH0FP[1:0] will select the active trigger or |

capture polarity for multi mode channel 0 input signals.
00: Multi mode channel 0 input signal's rising edge is the active signal for capture or trigger operation in slave mode. And multi mode channel 0 input signal will not be inverted.
01: Multi mode channel 0 input signal's falling edge is the active signal for capture or trigger operation in slave mode. And multi mode channel 0 input signal will be inverted.
10: Reserved.
11: Noninverted/both multi mode channel 0 input signal's edges.
This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 or 10.

### Multi mode channel 0 capture/compare value register (TIMERx_MCH0CV)

Address offset: 0x54
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MCH0VAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | MCH0VAL[15:0] | Capture/compare value of multi mode channel 0.<br>When multi mode channel 0 is configured in input mode, this bit-field indicates the counter value at the last capture event. And this bit-field is read-only.<br>When multi mode channel 0 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. |

### Control register 2 (TIMERx_CTL2)

Address offset: 0x74
Reset value: 0x0030 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | MCH0MSEL[1:0] | | Reserved | | | |
| | | | | | | | | | | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved |
|---|

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:20 | MCH0MSEL[1:0] | Multi mode channel 0 mode select<br>00: Independent mode, MCH0 is independent of CH0<br>01: Reserved<br>10: Reserved<br>11: Complementary mode, only the CH0 is valid for input, and the outputs of MCH0 and CH0 are complementary |
| 19:0 | Reserved | Must be kept at reset value. |

### TIMER alternate function control register 0 (TIMERx_AFCTL0)

Address offset: 0x8C

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1P | BRK0CMP0P | Reserved | | | | | | | | BRK0IN0P |
| | | | | | rw | rw | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | BRK0CMP1EN | BRK0CMP0EN | BRK0HPDFEN | Reserved | | | | | | | BRK0IN0EN |
| | | | | | rw | rw | rw | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | BRK0CMP1P | BREAK0 CMP1 input polarity<br>This bit is used to configure the CMP1 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: CMP1 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP1 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 25 | BRK0CMP0P | BREAK0 CMP0 input polarity<br>This bit is used to configure the CMP0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit. |

| | | |
|---|---|---|
| | | 0: CMP0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: CMP0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 24:17 | Reserved | Must be kept at reset value. |
| 16 | BRK0IN0P | BREAK0 BRKIN0 alternate function input polarity<br>This bit is used to configure the BRKIN0 input polarity, and the specific polarity is determined by this bit and the BRK0P bit.<br>0: BRKIN0 input signal will not be inverted (BRK0P =0, the input signal is active low; BRK0P =1, the input signal is active high)<br>1: BRKIN0 input signal will be inverted (BRK0P=0, the input signal is active high; BRK0P =1, the input signal is active low)<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 15:11 | Reserved | Must be kept at reset value. |
| 10 | BRK0CMP1EN | BREAK0 CMP1 enable<br>0: CMP1 input disabled<br>1: CMP1 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 9 | BRK0CMP0EN | BREAK0 CMP0 enable<br>0: CMP0 input disabled<br>1: CMP0 input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 8 | BRK0HPDFEN | BREAK0 HPDF input(hpdf_break[x], please refer to ***Table 45-2. HPDF internal signal***) enable<br>0: HPDF input disabled<br>1: HPDF input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | BRK0IN0EN | BREAK0 BRKIN0 alternate function input enable<br>0: BRKIN0 alternate function input disabled<br>1: BRKIN0 alternate function input enabled<br>This bit can be modified only when PROT[1:0] bit-field in TIMERx_CCHP register is 00. |

### Counter repetition register 1 (TIMERx_CREP1)

Address offset: 0x98

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CREP1[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CREP1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CREP1[31:0] | Counter repetition value 1<br>This bit-field is 32 bits and can be read on the fly.<br>This bit-field specifies the update event generation rate. Each time the repetition counter counts down to zero, an update event will be generated. The update rate of the shadow registers is also affected by this bit-field when these shadow registers are enabled.<br>**Note**: This bit-field just used with CREPSEL =1(in TIMERx_CFG register). |

### DMA configuration register (TIMERx_DMACFG)

Address offset: 0xE0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DMATC[5:0] | | | | | | Reserved | | DMATA[5:0] | | | | | |
| | | rw | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:8 | DMATC[5:0] | DMA transfer count<br>This field defines the times of accessing(R/W) the TIMERx_DMATB register by DMA.<br>6’b000000: transfer 1 time<br>6’b000001: transfer 2 times<br>…<br>6’b100101: transfer 38 times |

| | | |
|---|---|---|
| 7:6 | Reserved | Must be kept at reset value. |
| 5:0 | DMATA[5:0] | DMA transfer access start address<br>This field defines the start address of accessing the TIMERx_DMATB register by DMA. When the first access to the TIMERx_DMATB register is done, this bit-field specifies the address just accessed. And then the address of the second access to the TIMERx_DMATB register will be (start address + 0x4).<br>6'b000000: TIMERx_CTL0<br>6'b000001: TIMERx_CTL1<br>…<br>In a word: start address = TIMERx_CTL0 + DMATA*4 |

### DMA transfer buffer register (TIMERx_DMATB)

Address offset: 0xE4
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATB[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMATB[31:0] | DMA transfer buffer<br>When a read or write operation is assigned to this register, the register located at the address ranges from (start address) to (start address + transfer count * 4) will be accessed.<br>The transfer count is calculated by hardware, and ranges from 0 to DMATC. |

### Configuration register (TIMERx_CFG)

Address offset: 0xFC
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CCUSEL | CREPSEL | CHVSEL | OUTSEL |
| | | | | | | | | | | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | CCUSEL | Commutation control shadow register update select<br>This bit is valid only when the CCUC[2:0] bit-field are set to 100.<br>0: The shadow registers update when the counter generates an overflow/ underflow event.<br>1: The shadow registers update when the counter generates an overflow/ underflow event and the repetition counter value is zero. |
| 2 | CREPSEL | The counter repetition register select<br>This bit is used to select the counter repetition register.<br>0: The update event rate is depended to TIMERx_CREP0 register<br>1: The update event rate is depended to TIMERx_CREP1 register |
| 1 | CHVSEL | Write CHxVAL register selection bit<br>This bit-field is set and reset by software.<br>1: If the value to be written to the CHxVAL register is the same as the value of CHxVAL register, the write access is ignored.<br>0: No effect. |
| 0 | OUTSEL | The output value selection bit<br>This bit-field is set and reset by software.<br>1: If POEN bit and IOS bit are 0, the output is disabled.<br>0: No effect. |

# 24.5. Basic timer (TIMERx, x=5,6,50,51)

## 24.5.1. Overview

The basic timer module(TIMER5/6/50/51) has a 32-bit or 64-bit counter that can be used as an unsigned counter. The basic timer can be configured to generate a DMA request and a TRGO0 to connect to DAC.

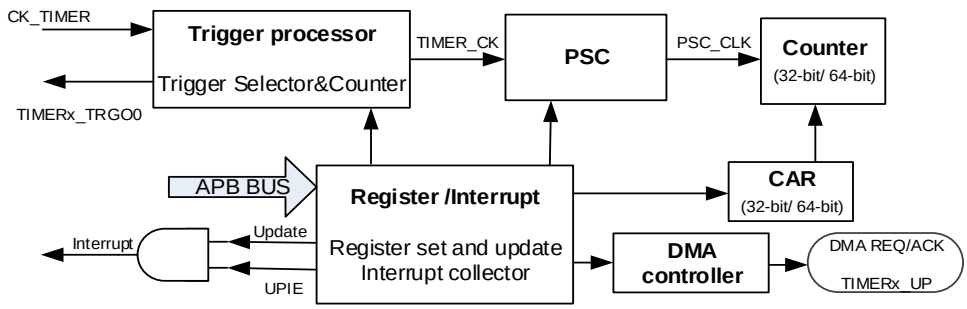
## 24.5.2. Characteristics

- Counter width: 32 bits (TIMER5/6), 64 bits (TIMER50/51).
- Source of count clock is internal clock only.
- Multiple counter modes: count up.
- Programmable prescaler: 16 bits. The factor can be changed ongoing.
- Auto reload function.
- Interrupt output or DMA request: update event.

## 24.5.3. Block diagram

*Figure 24-126. Basic timer block diagram* provides details on the internal configuration of the basic timer.

**Figure 24-126. Basic timer block diagram**



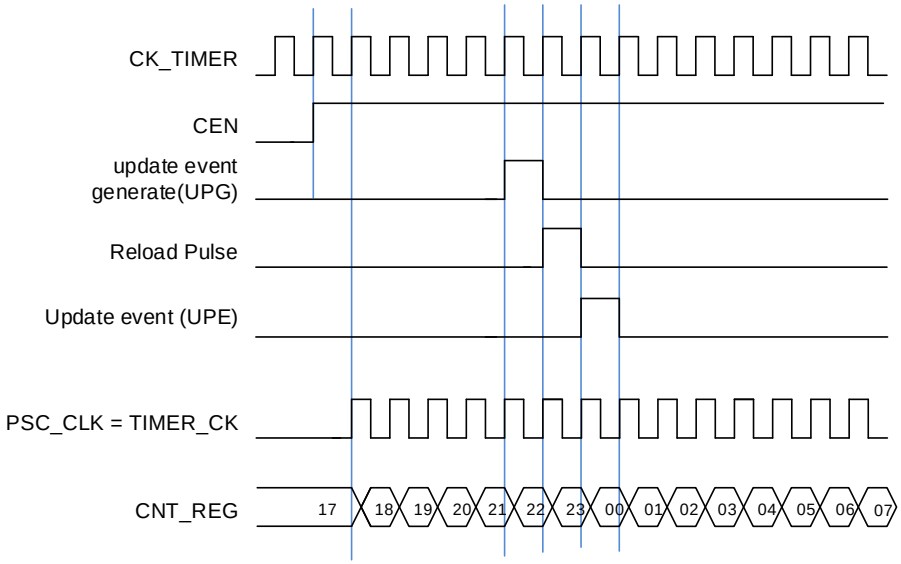

## 24.5.4. Function overview

### Clock selection

The basic TIMER can only be clocked by the internal timer clock CK_TIMER, which is from the source named CK_TIMER in RCU

The TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER used to drive the counter prescaler. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

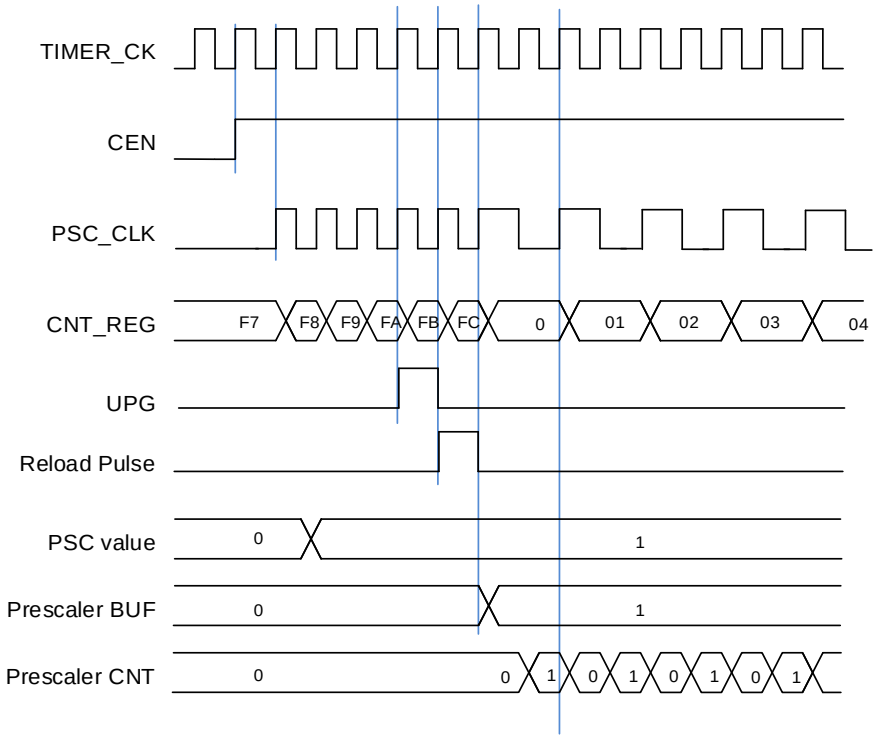
**Figure 24-127. Normal mode, internal clock divided by 1**




## Prescaler

The prescaler can divide the timer clock (TIMER_CK) to a counter clock (PSC_CLK) by any factor ranging from 1 to 65536. It is controlled by prescaler register (TIMERx_PSC) which can be changed ongoing, but it is adopted at the next update event.

**Figure 24-128. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter reload value, which is defined in the TIMERx_CAR/ TIMERx_CARL/ TIMERx_CARH register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. The update event is generated each time when counter overflows. The counting direction bit DIR in the TIMERx_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and an update event will be generated.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99 (TIMERx, x=5,6).

**Figure 24-129. Timing chart of up counting mode, PSC=0/2 (TIMERx, x=5,6)**

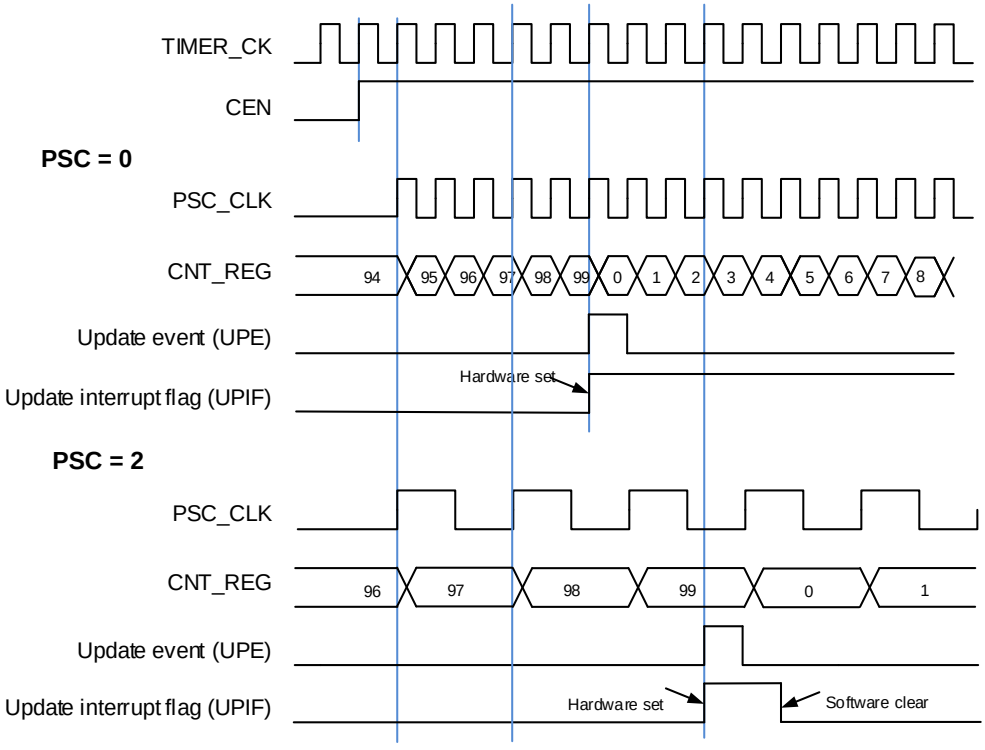
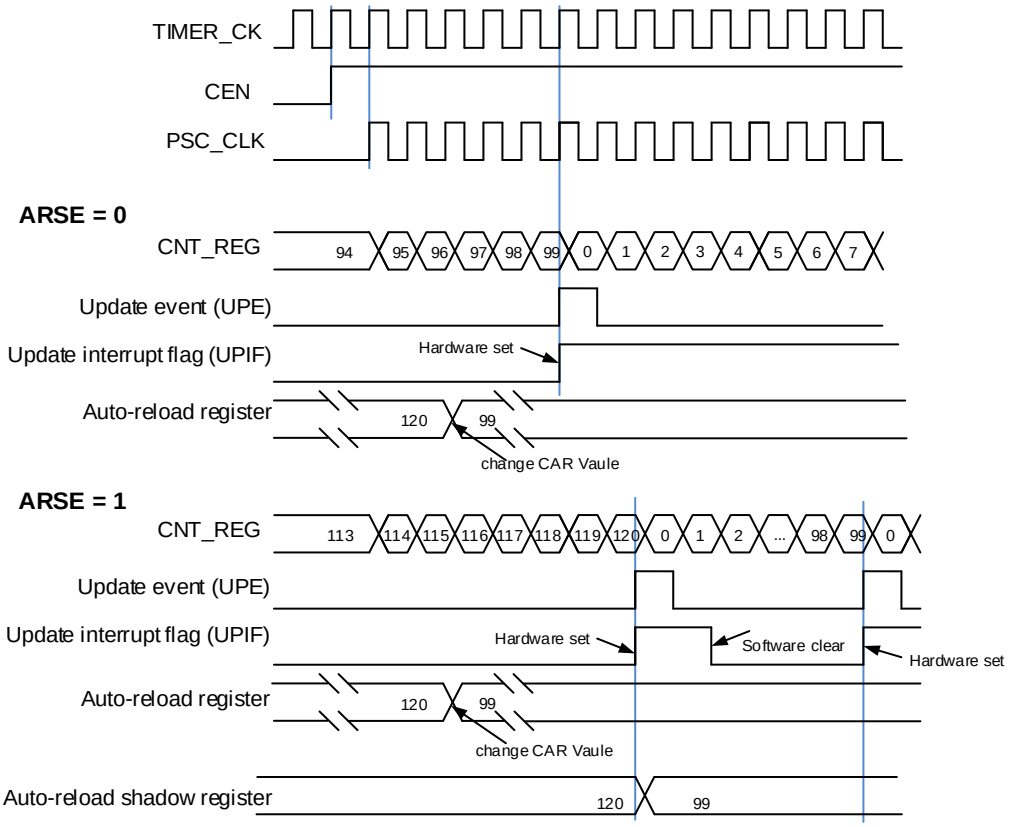



**Figure 24-130. Timing chart of up counting mode, change TIMERx_CAR ongoing**

**(TIMERx, x=5,6)**




## UPIF bit backup

The UPIF bit backup function is enabled by setting UPIFBUEN in the TIMERx_CTL0 register. The UPIF and UPIFBU bits are fully synchronized and without latency.

By using this function, the UPIF bit in the TIMERx_INTF register will be backuped to the UPIFBU bit in the TIMERx_CNT register. This can avoid conflicts when reading the counter and interrupt processing.

## Timer debug mode

When the Cortex®-M7 is halted, and the TIMERx_HOLD configuration bit in DBG_CTL register set to 1, the TIMERx counter stops.

## 24.5.5. Registers definition (TIMERx, x=5,6,50,51)

TIMER5 base address: 0x4000 1000

TIMER6 base address: 0x4000 1400

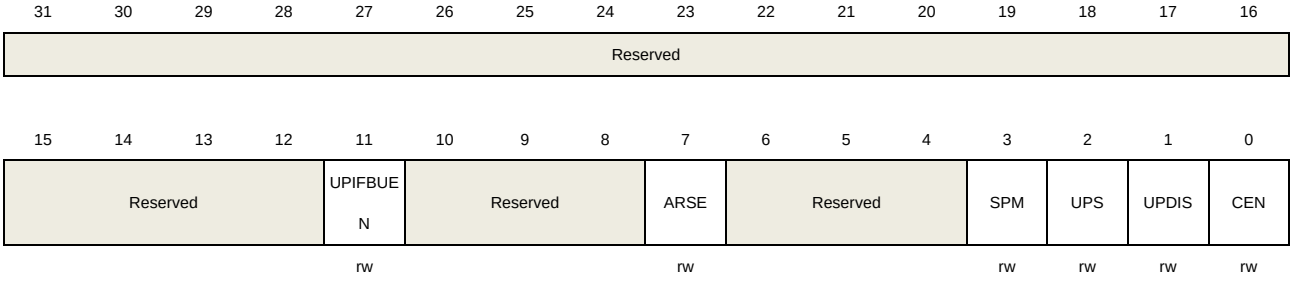
TIMER50 base address: 0x4000 F000

TIMER51 base address: 0x4000 F400

### Control register 0 (TIMERx_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | UPIFBUEN | UPIF bit backup enable<br>0: Backup disable. UPIF bit is not backuped to UPIFBU bit in TIMERx_CNT/ TIMERx_CNTH register.<br>1: Backup enabled. UPIF bit is backuped to UPIFBU bit in TIMERx_CNT/ TIMERx_CNTH register. |
| 10:8 | Reserved | Must be kept at reset value. |
| 7 | ARSE | Auto-reload shadow enable<br>0: The shadow register for TIMERx_CAR/ TIMERx_CARL/ TIMERx_CARH register is disabled<br>1: The shadow register for TIMERx_CAR/ TIMERx_CARL/ TIMERx_CARH register is enabled |
| 6:4 | Reserved | Must be kept at reset value. |
| 3 | SPM | Single pulse mode.<br>0: Single pulse mode is disabled. Counter continues after an update event.<br>1: Single pulse mode is enabled. The CEN bit is cleared by hardware and the counter stops at next update event. |
| 2 | UPS | Update source |

This bit is used to select the update event sources by software.

0: When enabled, any of the following events generates an update interrupt or a DMA request:

- The UPG bit is set
- The counter generates an overflow event
- The slave mode controller generates an update event.

1: When enabled, only counter overflow generates an update interrupt or a DMA request.

1 UPDIS Update disable.

This bit is used to enable or disable the update event generation.

0: Update event enable. The update event is generated and the buffered registers are loaded with their preloaded values when one of the following events occurs:

- The UPG bit is set
- The counter generates an overflow event
- The slave mode controller generates an update event.

1: Update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or the slave mode controller generates a hardware reset event.

0 CEN Counter enable

0: Counter disable

1: Counter enable

The CEN bit must be set by software when timer works in external clock mode, pause mode or decoder mode. While in event mode, the hardware can set the CEN bit automatically.

### Control register 1 (TIMERx_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | MMC0[2:0] | | | Reserved | | | |
| | | | | | | | | | rw | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6:4 | MMC0[2:0] | Master mode control 0<br>These bits control the selection of TRGO0 signal, which is sent by master timer to slave timer for synchronization function. |

| | | |
|---|---|---|
| | | 000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO0 pulse occurs. And in the latter case, the signal on TRGO0 is delayed compared to the actual reset.<br>001: Enable. This mode is used to start several timers at the same time or control a slave timer to be enabled in a period. In this mode, the master mode controller selects the counter enable signal as TRGO0. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO0 output, except if the master-slave mode is selected.<br>010: Update. In this mode, the master mode controller selects the update event as TRGO0.<br>100~111: Reserved. |
| 3:0 | Reserved | Must be kept at reset value. |

### Interrupt enable register (TIMERx_DMAINTEN)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | UPDEN | Reserved | | | | | | | UPIE |
| | | | | | | | rw | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | UPDEN | Update DMA request enable<br>0: Disabled<br>1: Enabled |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | UPIE | Update interrupt enable<br>0: Disabled<br>1: Enabled |

### Interrupt flag register (TIMERx_INTF)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | UPIF |
| | | | | | | | | | | | | | | | rc_w0 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | UPIF | Update interrupt flag<br>This bit is set by hardware when an update event occurs and cleared by software.<br>0: No update interrupt occurred<br>1: Update interrupt occurred |

### Software event generation register (TIMERx_SWEVG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | UPG |
| | | | | | | | | | | | | | | | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | UPG | This bit can be set by software, and automatically cleared by hardware. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time.<br>0: No generate an update event<br>1: Generate an update event |

### Counter register (TIMERx_CNT) (TIMERx, x=5,6)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[31] | CNT[30:16] | | | | | | | | | | | | | | |
| UPIFBU | | | | | | | | | | | | | | | |
| rw/ r | rw | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

**UPIFBUEN = 0:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CNT[31:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

**UPIFBUEN = 1:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register, and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |
| 30 | CNT[30:0] | This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter. |

### Counter low register (TIMERx_CNTL) (TIMERx, x=50,51)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNTL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNTL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CNTL[31:0] | This bit-field indicates the current counter low value. Writing to this bit-field can change the value of the counter. |

### Prescaler register (TIMERx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSC[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | PSC[15:0] | Prescaler value of the counter clock<br>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event. |

### Counter auto reload register (TIMERx_CAR) (TIMERx, x=5,6)

Address offset: 0x2C
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CARL[31:0] | Counter auto reload value<br>This bit-field specifies the auto reload value of the counter. |

### Counter auto reload low register (TIMERx_CARL) (TIMERx, x=50,51)

Address offset: 0x2C
Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARLL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARLL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CARLL[31:0] | Counter auto reload low value<br>This bit-field specifies the auto reload low value of the counter. |

### Counter high register (TIMERx_CNTH) (TIMERx, x=50,51)

Address offset: 0xD0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNTH[31] / UPIFBU | CNTH[63:48] | | | | | | | | | | | | | | |
| rw/ r | rw | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNTH[47:32] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

**UPIFBUEN = 0:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CNTH[63:32] | This bit-field indicates the current counter high value. Writing to this bit-field can change the value of the counter. |

**UPIFBUEN = 1:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | UPIFBU | UPIF bit backup<br>This bit is a backup of UPIF bit in TIMERx_INTF register, and read-only. This bit is only valid when UPIFBUEN = 1. If the UPIFBUEN =0, this bit is reserved and read the result is 0. |
| 30:0 | CNTH[62:32] | This bit-field indicates the current counter high value. Writing to this bit-field can change the value of the counter. |

### Counter auto reload high register 1(TIMERx_CARH)(TIMERx, x=50,51)

Address offset: 0xD4

Reset value: 0xFFFF FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARLH[63:48] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARLH[47:32] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CARLH[63:32] | Counter auto reload high value<br>This bit-field specifies the auto reload high value of the counter. |

# 25. Universal synchronous / asynchronous receiver / transmitter (USART)

## 25.1. Overview

The Universal Synchronous / Asynchronous Receiver / Transmitter (USART) provides a flexible serial data exchange interface. Data frames can be transferred in full duplex or half duplex mode, synchronously or asynchronously through this interface. A programmable baud rate generator divides the UCLK (CK_APBx, CK_AHB, CK_LXTAL or CK_IRC64MDIV) to produces a dedicated wide range baudrate clock for the USART transmitter and receiver.

Besides the standard asynchronous receiver and transmitter mode, the USART implements several other types of serial data exchange modes, such as IrDA (infrared data association) SIR mode, smartcard mode, LIN (local interconnection network) mode, half-duplex mode and synchronous mode. It also supports multiprocessor communication mode, and hardware flow control protocol (CTS / RTS). The data frame can be transferred from LSB or MSB bit. The polarity of the TX / RX pins can be configured independently and flexibly.

All USARTs support DMA function for high-speed data communication.

## 25.2. Characteristics

- NRZ standard format.
- Asynchronous, full duplex communication.
- Half duplex single wire communications.
- Receive / Transmit FIFO function.
- Address match feature in the receiver to reduce address mark wakeup ISR overhead.
- Dual clock domain:
  - Asynchronous PCLK and USART clock.
  - Baud rate programming independent from the UCLK reprogramming.
- Programmable baud-rate generator allowing speed up to 37.5 MBits/s when the clock frequency is 300 MHz and oversampling is by 8.
- Fully programmable serial interface characteristics:
  - A data word (7 / 8 / 9 or 10 bits) LSB or MSB first.
  - Even, odd or no-parity bit generation / detection.
  - 0.5, 1, 1.5 or 2 stop bit generation.
- Swappable Tx / Rx pin.
- Configurable data polarity.
- Hardware modem operations (CTS / RTS) and RS485 drive enable.
- Configurable multibuffer communication using centralized DMA.
- Separate enable bits for transmitter and receiver.
- Parity control:

  - Transmits parity bit.
  - Checks parity of received data byte.
- LIN break generation and detection.
- IrDA support.
- Synchronous mode and transmitter clock output for synchronous transmission.
- ISO 7816-3 compliant smartcard interface:
  - Character mode (T = 0).
  - Block mode (T = 1).
  - Direct and inverse convention.
- Multiprocessor communication:
  - Enter into mute mode if address match does not occur.
  - Wake up from mute mode by idle line or address match detection.
- Support for ModBus communication:
  - Timeout feature.
  - CR / LF character recognition.
- Wake up from deep-sleep mode:
  - By standard RBNE interrupt.
  - By WUF interrupt.
- Various status flags:
  - Flags for transfer detection: Receive buffer not empty (RBNE), Receive FIFO full (RFF), Receive FIFO empty (RFE), Receive FIFO threshold reached(RFT), Transmit buffer empty (TBE), transfer complete (TC), Transmit FIFO not full(TFNF), Transmit FIFO empty(TFE), Transmit FIFO threshold reached(TFTIF).
  - Flags for error detection: overrun error (ORERR), noise error (NERR), frame error (FERR) and parity error (PERR).
  - Flag for hardware flow control: CTS changes (CTSF).
  - Flag for LIN mode: LIN break detected (LBDF).
  - Flag for multiprocessor communication: IDLE frame detected (IDLEF).
  - Flag for ModBus communication: address/character match (AMF) and receiver timeout (RTF).
  - Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF).
  - Wakeup from deep-sleep mode flag.
  - Interrupt occurs at these events when the corresponding interrupt enable bits are set.

While USART0 / USART1 / USART2 / USART5 is fully implemented, UART3 / UART4 / UART6 / UART7 is only partially implemented with the following features not supported.

- Smartcard mode.
- IrDA SIR ENDEC block.
- LIN mode.
- Dual clock domain and wakeup from deep-sleep mode.
- Receiver timeout interrupt.
- ModBus communication.
- Synchronous mode.

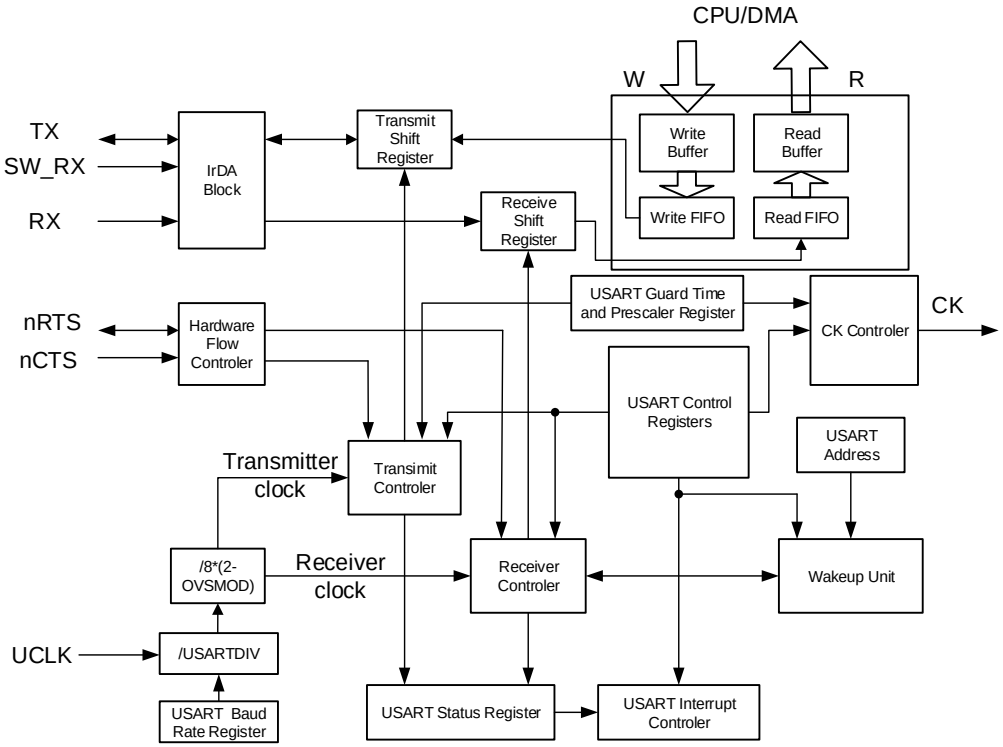
## 25.3. Function overview

The interface is externally connected to another device by the main pins listed in ***Table 25-1. Description of USART important pins***.

**Table 25-1. Description of USART important pins**

| Pin | Type | Description |
|---|---|---|
| RX | Input | Receive Data |
| TX | Output I/O (single-wire / smartcard mode) | Transmit Data. High level When enabled but nothing to be transmitted |
| CK | Output | Serial clock for synchronous communication |
| nCTS | Input | Clear to send in Hardware flow control mode |
| nRTS | Output | Request to send in Hardware flow control mode |

**Figure 25-1. USART module block diagram**

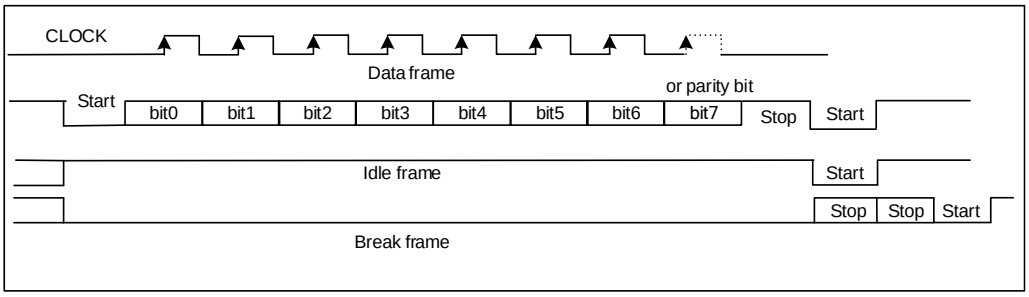



### 25.3.1. USART frame format

The USART frame starts with a start bit and ends up with a number of stop bits. The length of the data frame is configured by the WL0 bit and WL1 bit in the USART_CTL0 register. The last data bit can be used as parity check bit by setting the PCEN bit of in USART_CTL0 register. When the WL1 bit and WL0 bit are reset, the parity bit is the 7th bit. When the WL1 bit is reset and WL0 bit is set, the parity bit is the 8th bit. When WL1 bit is set and the WL0 bit is reset, the parity bit is the 6th bit. When WL1 bit is set and the WL0 bit is set, the parity bit is the 9th bit. The method of calculating the parity bit is selected by the PM bit in USART_CTL0

register.

**Figure 25-2. USART character frame (8 bits data and 1 stop bit)**




In transmission and reception, the number of stop bits can be configured by the STB[1:0] bits in the USART_CTL1 register.

**Table 25-2. Configuration of stop bits**

| STB[1:0] | stop bit length (bit) | usage description |
|---|---|---|
| 00 | 1 | Default value |
| 01 | 0.5 | Smartcard mode for receiving |
| 10 | 2 | Normal USART and single-wire modes |
| 11 | 1.5 | Smartcard mode for transmitting and receiving |

In an idle frame, all the frame bits are logic 1. The frame length is equal to the normal USART frame.

The break frame structure is a number of low bits followed by the configured number of stop bits. The transfer speed of a USART frame depends on the frequency of the UCLK, the configuration of the baud rate generator and the oversampling mode.

### 25.3.2. Baud rate generation

The baud-rate divider is a 16-bit number which consists of a 12-bit integer and a 4-bit fractional part. The number formed by these two values is used by the baud rate generator to determine the bit period. Having a fractional baud-rate divider allows the USART to generate all the standard baud rates.

The baud-rate divider (USARTDIV) has the following relationship with the UCLK:

In case of oversampling by 16, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}} \quad (25\text{-}1)$$

In case of oversampling by 8, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{8 \times \text{Baud Rate}} \quad (25\text{-}2)$$

For example, when oversampled by 16:

1. Get USARTDIV by caculating the value of USART_BAUD:
   If USART_BAUD = 0x21D, then INTDIV = 33 (0x21), FRADIV = 13 (0xD).

USARTDIV=33 + 13 / 16 = 33.81.

2. Get the value of USART_BAUD by calculating the value of USARTDIV:
   If USARTDIV = 30.37, then INTDIV = 30 (0x1E).
   16*0.37 = 5.92, the nearest integer is 6, so FRADIV = 6 (0x6).
   USART_BAUD = 0x1E6.

   **Note:** If the roundness of FRADIV is 16 (overflow), the carry must be added to the integer part.

### 25.3.3. USART transmitter

If the transmit enable bit (TEN) in USART_CTL0 register is set, when the transmit data buffer is not empty, the transmitter shifts out the transmit data frame through the TX pin. The polarity of the TX pin can be configured by the TINV bit in the USART_CTL1 register. Clock pulses can output through the CK pin.

After the TEN bit is set, an idle frame will be sent. The TEN bit should not be cleared while the transmission is ongoing.

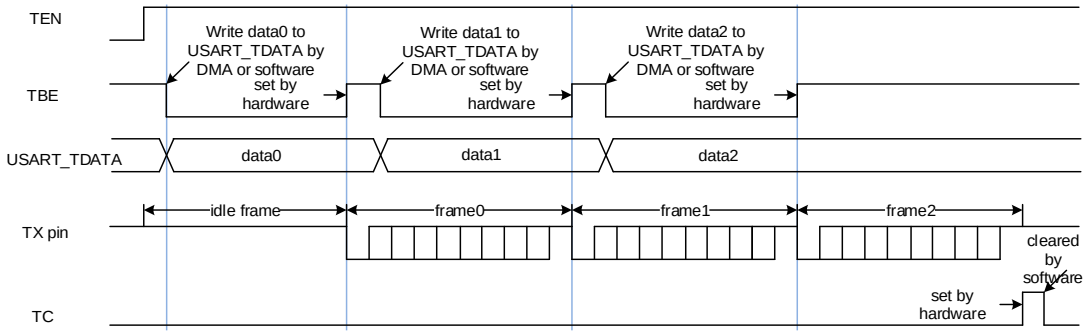
After power on, the TBE bit is high by default. Data can be written to the USART_TDATA when the TBE bit in the USART_STAT register is asserted. The TBE bit is cleared by writing USART_TDATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written to the USART_TDATA register while a transmission is ongoing, it will be firstly stored in the transmit buffer, and transferred to the transmit shift register after the current transmission is done. If a data is written to the USART_TDATA register while no transmission is ongoing, the TBE bit will be cleared and set soon, because the data will be transferred to the transmit shift register immediately.

If a frame is transmitted and the TBE bit is asserted, the TC bit of the USART_STAT register will be set. An interrupt will be generated if the corresponding interrupt enable bit (TCIE) is set in the USART_CTL0 register.

The USART transmit procedure is shown in ***Figure 25-3. USART transmit procedure***. The software operating process is as follows:

1. Write the WL0 bit and WL1 bit in USART_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART_CTL1 to configure the number of stop bits.
3. Enable DMA (DENT bit) in USART_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART_BAUD.
5. Set the UEN bit in USART_CTL0 to enable the USART.
6. Set the TEN bit in USART_CTL0.
7. Wait for the TBE being asserted.
8. Write the data to the USART_TDATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

**Figure 25-3. USART transmit procedure**




It is necessary to wait for the TC bit to be asserted before disabling the USART or entering the power saving mode. The TC bit can be cleared by set the TCC bit in USART_INTC register.

The break frame is sent when the SBKCMD bit is set, and SBKCMD bit is reset after the transmission.

### 25.3.4. USART receiver

After power on, the USART receiver can be enabled by the following procedure:

1. Write the WL0 bit and WL1 bit in USART_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART_CTL1.
3. Enable DMA (DENR bit) in USART_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART_BAUD.
5. Set the UEN bit in USART_CTL0 to enable the USART.
6. Set the REN bit in USART_CTL0.

After being enabled, the receiver receives a bit stream after a valid start pulse has been detected. Detection on noisy error, parity error, frame error and overrun error is performed during the reception of a frame.

When a frame is received, the RBNE bit in USART_STAT is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART_CTL0 register. The status of the reception are stored in the USART_STAT register.

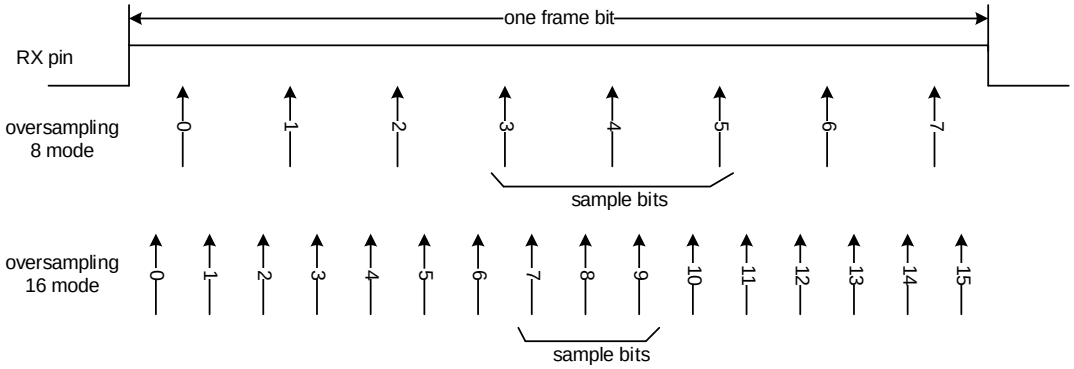
The software can get the received data by reading the USART_RDATA register directly, or through DMA. The RBNE bit is cleared by a read operation on the USART_RDATA register, whatever it is performed by software directly, or through DMA.

The REN bit should not be disabled when reception is ongoing, or the current frame will be lost.

By default, the receiver gets three samples to evaluate the value of a frame bit. If the oversampling 8 mode is enabled, the 3rd, 4th and 5th samples are used, while in the oversampling 16 mode, the 7th, 8th, and 9th samples are used. If two or more samples of a frame bit is 0, the frame bit is confirmed as a 0, else 1. If the value of the three samples of any bit are not the same, whatever it is a data bit, parity bit or stop bit, a noisy error (NERR) status will be generated for the frame. An interrupt will be generated, If the ERRIE bit in

USART_CTL2 register is set. If the OSB bit in USART_CTL2 register is set, the receiver gets only one sample to evaluate a bit value. In this situation, no noisy error will be detected.

**Figure 25-4. Oversampling method of a receive frame bit (OSB = 0)**




If the parity check function is enabled by setting the PCEN bit in the USART_CTL0 register, the receiver calculates the expected parity value while receiving a frame. The received parity bit will be compared with this expected value. If they are not the same, the parity error (PERR) bit in USART_STAT register will be set. An interrupt is generated, if the PERRIE bit in USART_CTL0 register is set.

If the RX pin is evaluated as 0 during a stop bit, the frame error (FERR) bit in USART_STAT register will be set. An interrupt is generated, If the ERRIE bit in USART_CTL2 register is set. According to the configuration of the stop bit, there are the following situations:

- 0.5 stop bit: When 0.5 stop bit, stop bit is not sampled.
- 1 stop bit: When 1 stop bit, sampling in the middle of stop bit.
- 1.5 stop bits: When 1.5 stop bits, the 1.5 stop bits are divided into 2 parts: the 0.5 stop bit part is not sampled and sampling in the middle of 1 stop bit.
- 2 stop bits: When 2 stop bits, if a frame error is detected during the first stop bit, the frame error flag is set, the second stop bit is not checked frame error. If no frame error is detected during the first stop bit, then continue to check the second stop bit for frame error.

When a frame is received, if the RBNE bit is not cleared yet, the last frame will not be stored in the receive data buffer. The overrun error (ORERR) bit in USART_STAT register will be set. An interrupt is generated, if the ERRIE bit in USART_CTL2 register is set, or if the RBNEIE is set.

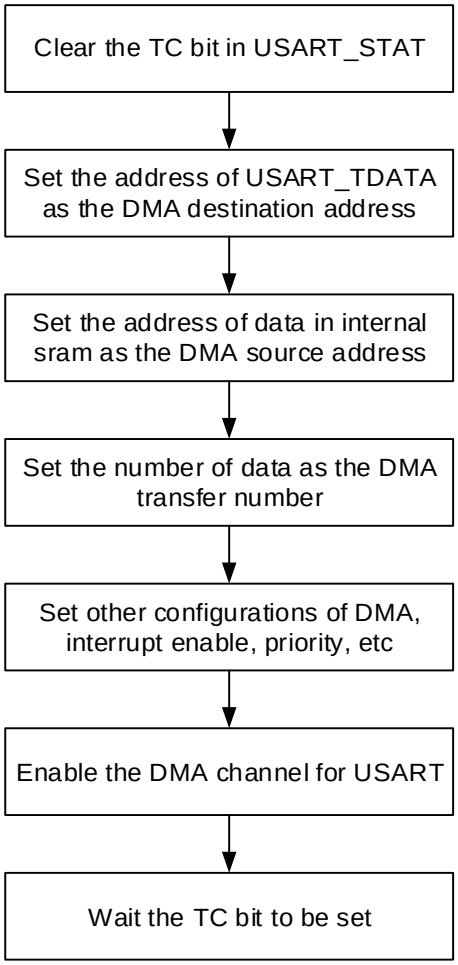
If a noise error (NERR), parity error (PERR), frame error (FERR) or overrun error (ORERR) occurs during reception, NERR, PERR, FERR or ORERR will be set at the same time with RBNE. If the receive DMA is not enabled, when the RBNE interrupt occurs, software need to check whether there is a noise error, parity error, frame error or overrun error.

### 25.3.5. Use DMA for data buffer access

To reduce the burden of the processor, DMA can be used to access the transmitting and receiving data buffer. The DENT bit in USART_CTL2 is used to enable the DMA transmission, and the DENR bit in USART_CTL2 is used to enable the DMA reception.

When DMA is used for USART transmission, DMA transfers data from internal SRAM to the transmit data buffer of the USART. The configuration step are shown in ***Figure 25-5. Configuration step when using DMA for USART transmission***.

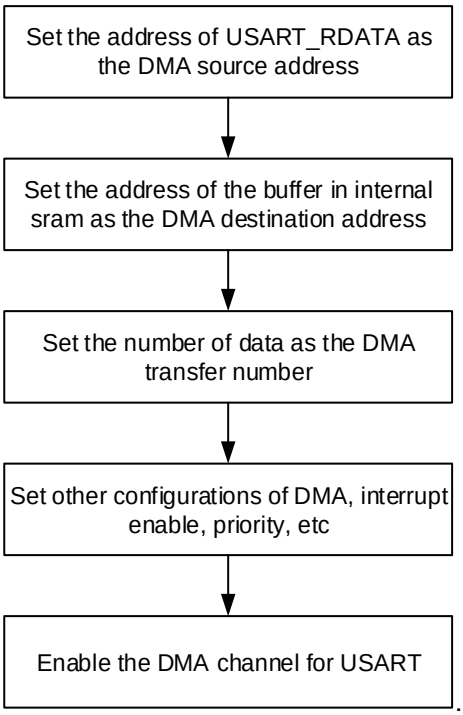
**Figure 25-5. Configuration step when using DMA for USART transmission**




After all of the data frames are transmitted, the TC bit in USART_STAT is set. An interrupt occurs if the TCIE bit in USART_CTL0 is set.

When DMA is used for USART reception, DMA transfers data from the receive data buffer of the USART to the internal SRAM. The configuration steps are shown in ***Figure 25-6. Configuration step when using DMA for USART reception***. If the ERRIE bit in USART_CTL2 is set, interrupts can be generated by the error status bits (FERR, ORERR and NERR) in USART_STAT.

**Figure 25-6. Configuration step when using DMA for USART reception**

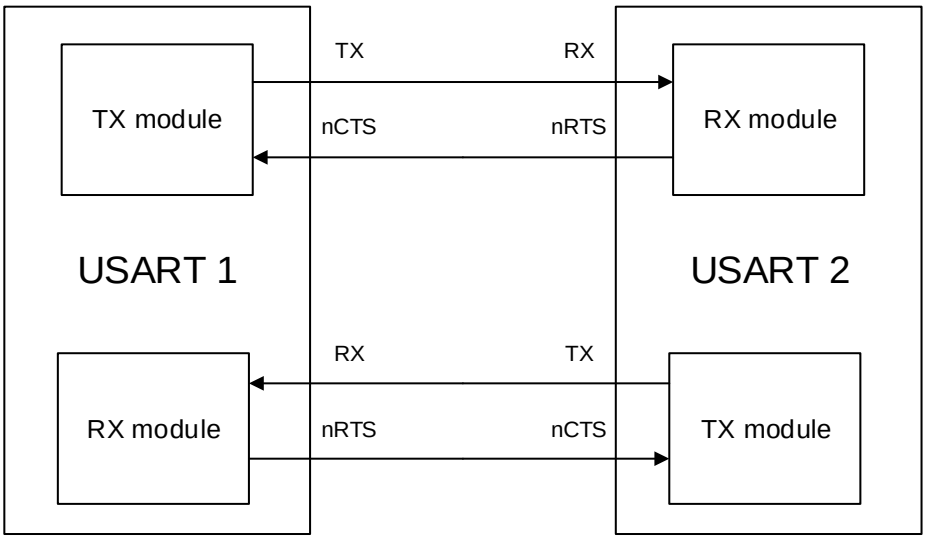



When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt can be generated in the DMA module.

### 25.3.6. Hardware flow control

The hardware flow control function is realized by the nCTS and nRTS pins. The RTS flow control is enabled by writing '1' to the RTSEN bit in USART_CTL2 and the CTS flow control is enabled by writing '1' to the CTSEN bit in USART_CTL2.

**Figure 25-7. Hardware flow control between two USARTs**



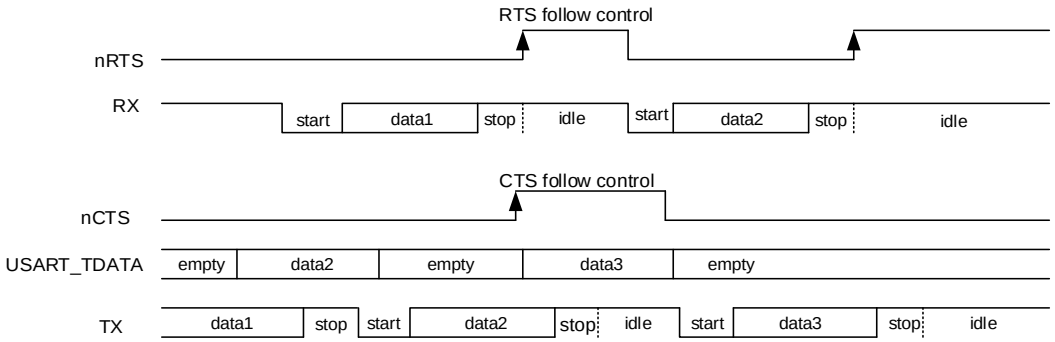
### RTS flow control

The USART receiver outputs the nRTS, which reflects the status of the receive buffer. When data frame is received, the nRTS signal goes high to prevent the transmitter from sending next frame. The nRTS signal keeps high when the receive buffer is full.

### CTS flow control

The USART transmitter monitors the nCTS input pin to decide whether a data frame can be transmitted. If the TBE bit in USART_STAT is '0' and the nCTS signal is low, the transmitter transmits the data frame. When the nCTS signal goes high during a transmission, the transmitter stops after the current transmission is accomplished.

**Figure 25-8. Hardware flow control**




### RS485 Driver Enable

The driver enable feature, which is enabled by setting bit DEM in the USART_CTL2 control register, allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time, which is programmed using the DEA [4:0] bits field in the USART_CTL0 control register, is the time between the activation of the DE signal and the beginning of the START bit. The de-assertion time, which is programmed using the DED [4:0] bits field in the USART_CTL0 control register, is the time between the end of the last stop bit and the de-activation of the DE signal. The polarity of the DE signal can be configured using the DEP bit in the USART_CTL2 control register.

## 25.3.7. Multi-processor communication

In multiprocessor communication, several USARTs are connected as a network. It will be a big burden for a device to monitor all of the messages on the RX pin. To reduce the burden of a device, software can put an USART module into a mute mode by writing 1 to the MMCMD bit in USART_CMD register.

If a USART is in mute mode, all of the receive status bits cannot be set. The USART can also be wake up by hardware by one of the two methods: idle frame method and address match method.

The idle frame wake up method is selected by default. If the RWU bit is reset, an idle frame

is detected on the RX pin, the IDLEF bit in USART_STAT will be set. If the RWU bit is set, an idle frame is detected on the RX pin, the hardware clears the RWU bit and exits the mute mode. When it is woken up by an idle frame, the IDLEF bit in USART_STAT will not be set.

When the WM bit of in USART_CTL0 register is set, the MSB bit of a frame is detected as the address flag. If the address flag is high, the frame is treated as an address frame. If the address flag is low, the frame is treated as a data frame. If the LSB 4 or 7 bits, which are configured by the ADDM0 bit of the USART_CTL1 register or ADDM1 bit of the USART_CTL2, of an address frame is the same as the ADDR0 bits in the USART_CTL1 register or ADDR1 bits in the USART_CTL2 register, the hardware will clear the RWU bit and exits the mute mode. The RBNE bit will be set when the frame that wakes up the USART. The status bits are available in the USART_STAT register. If the LSB 4/7 bits of an address frame defers from the ADDR0 bits in the USART_CTL1 register or ADDR1 bits in the USART_CTL2 register, the hardware sets the RWU bit and enters mute mode automatically. In this situation, the RBNE bit is not set.

If the PCEN bit in USART_CTL0 is set, the MSB bit will be checked as the parity bit, and the bit preceding the MSB bit is detected as the address bit. If the ADDM0 or ADDM1 bit is set and the receive frame is a 8bit data, the LSB 7 bits will be compared with ADDR0[6:0] or ADDR1[6:0]. If the ADDM0 or ADDM1 bit is set and the receive frame is a 9bit data, the LSB 8 bits will be compared with ADDR0[7:0] or ADDR1[7:0].

Match address operation function in the same way for both ADDR0 and ADDR1, Note that the the position of the address mark is the same as the Parity Bit When parity is enabled for 8 bit and 9 bit data formats.

If only one of AMEN0 and AMEN1 bit is set, a match address is compared only with the associate ADDR0 or ADDR1 and data is transferred to the receive data buffer only on a match.

If AMEN0 and AMEN1 are all set, a match address is compared with both ADDR0 and ADDR1 and data is transferred only on a match with either ADDR0 or ADDR1. So, a second match adderss can be used for broadcast or general call address to serial bus.

**Note:** If the MEN bit is set, the WM bit is reset and the RWU bit is reset, an idle frame is detected on the RX pin, the IDLEF bit will be set. If the RWU bit is set, the IDLEF is not set. Otherwise, AMEN0 or AMEN1 must be set when using address match method to wakeup USART from mute mode.

### 25.3.8. LIN mode

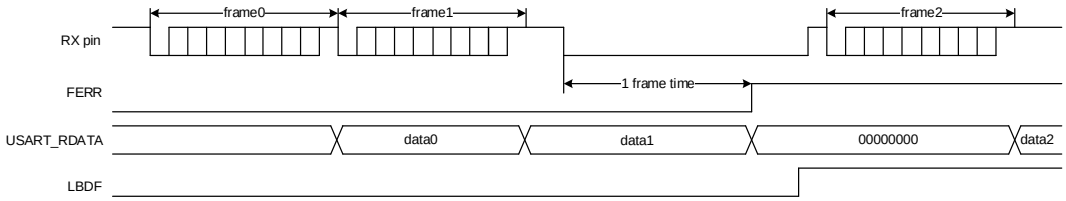
The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN, STB[1:0] bit in USART_CTL1 and the SCEN, HDEN, IREN bits in USART_CTL2 should be cleared in LIN mode.

When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. The data bits length can only be 8. And the break frame is 13-bit ’0’, followed by 1 stop bit.

The break detection function is totally independent of the normal USART receiver. So a break frame can be detected during the idle state or during a frame. The expected length of a break frame can be selected by configuring LBLEN in USART_CTL1. When the RX pin is detected at low state for a time that is equal to or longer than the expected break frame length (10 bits when LBLEN=0, or 11 bits when LBLEN=1), the LBDF bit in USART_STAT is set. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.

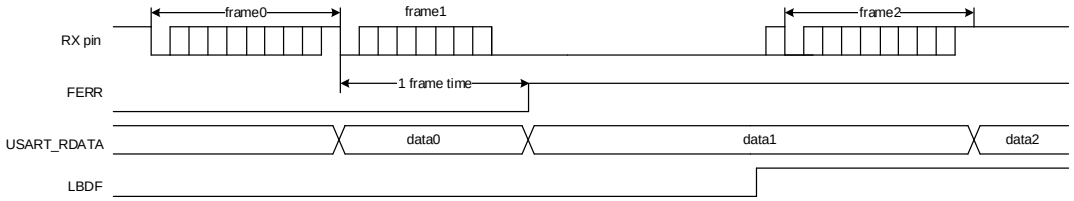
As shown in ***Figure 25-9. Break frame occurs during idle state***, if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all '0' frame, with an asserted FERR status.

**Figure 25-9. Break frame occurs during idle state**




As shown in ***Figure 25-10. Break frame occurs during a frame***, if a break frame occurs during a frame on the RX pin, the FERR status will be asserted for the current frame.

**Figure 25-10. Break frame occurs during a frame**




## 25.3.9. Synchronous mode

The USART can be used for full-duplex synchronous serial communications only in master mode, by setting the CKEN bit in USART_CTL1. The LMEN bit in USART_CTL1 and SCEN, HDEN, IREN bits in USART_CTL2 should be cleared in synchronous mode. The CK pin is the clock output of the synchronous USART transmitter, and can be only activated when the TEN bit is enabled. No clock pulse will be sent through the CK pin during the transmission of the start bit and stop bit. The CLEN bit in USART_CTL1 can be used to determine whether the clock is output or not during the last (address flag) bit transmission. The clock output is also not activated during idle and break frame sending. The CPH bit in USART_CTL1 can be used to determine whether data is captured on the first or the second clock edge. The CPL bit in USART_CTL1 can be used to configure the clock polarity in the USART synchronous idle state.

The CPL, CPH and CLEN bits in USART_CTL1 determine the waveform on the CK pin. Software can only change them when the USART is disabled (UEN = 0).

The clock is synchronized with the data transmitted. The receiver in synchronous mode samples the data on the transmitter clock without any oversampling.

**Figure 25-11. Example of USART in synchronous mode**

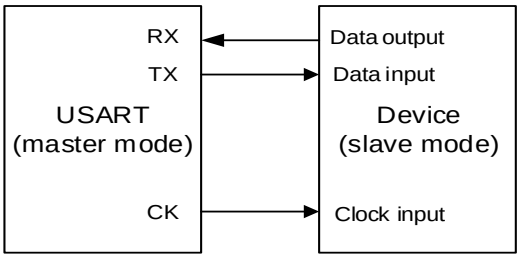
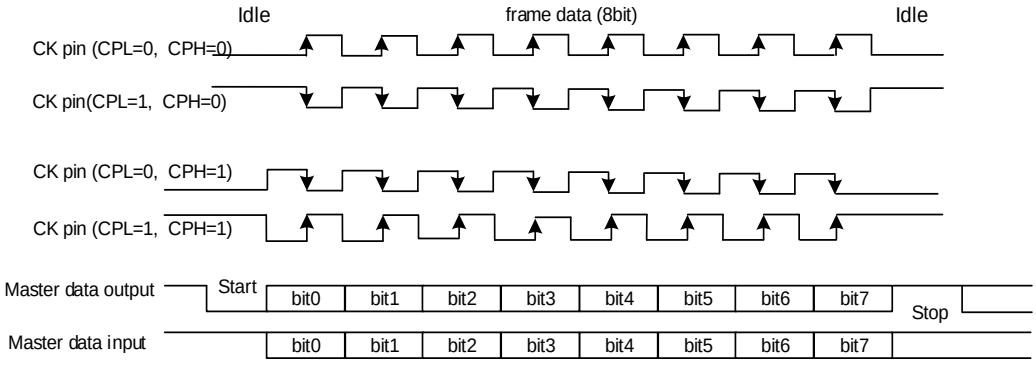



**Figure 25-12. 8-bit format USART synchronous waveform (CLEN = 1)**

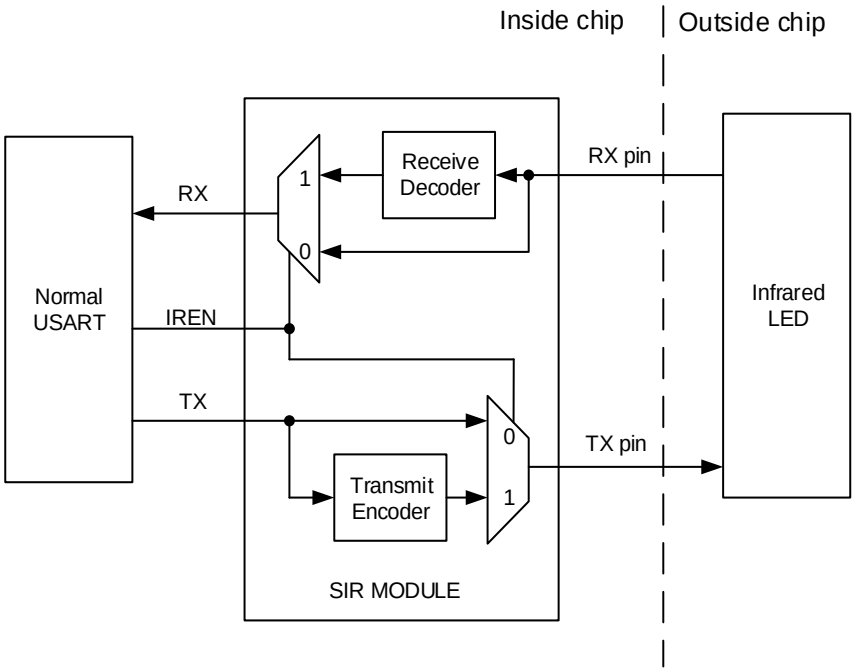



## 25.3.10. IrDA SIR ENDEC mode

The IrDA mode is enabled by setting the IREN bit in USART_CTL2. The LMEN, STB[1:0], CKEN bits in USART_CTL1 and HDEN, SCEN bits in USART_CTL2 should be cleared in IrDA mode.

In IrDA mode, the USART transmission data frame is modulated in the SIR transmit encoder and transmitted to the infrared LED through the TX pin. The SIR receive decoder receives the modulated signal from the infrared LED through the RX pin, and puts the demodulated data frame to the USART receiver. The baud rate should not be larger than 115200 for the encoder.

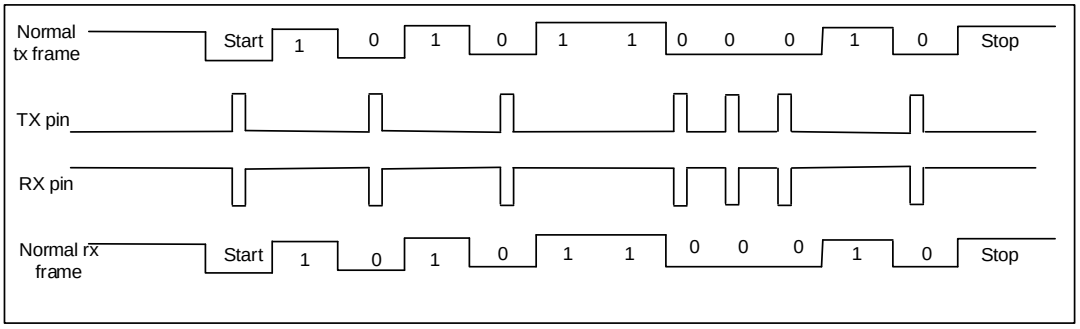
**Figure 25-13. IrDA SIR ENDEC module**




In IrDA mode, the polarity of the TX and RX pins is different. The TX pin is usually at low state, while the RX pin is usually at high state. The IrDA pins keep stable to represent the logic '1', while an infrared light pulse on the IrDA pins (a Return to Zero signal) represents the logic '0'. The pulse width should be 3 / 16 of a bit period. The IrDA could not detect any pulse if the pulse width is less than 1 PSC clock. While it can detect a pulse by chance if the pulse width is greater than 1 but smaller than 2 times of PSC clock.

Because the IrDA is a half-duplex protocol, the transmission and the reception should not be carried out at the same time in the IrDA SIR ENDEC block.

**Figure 25-14. IrDA data modulation**




The SIR sub module can work in low power mode by setting the IRLP bit in USART_CTL2. The transmit encoder is driven by a low speed clock, which is divided from the PCLK. The division ratio is configured by the PSC[7:0] bits in USART_GP register, USART BAUD register needs to be configured as 16*PSC[7:0]. The pulse width on the TX pin is 3 cycles of this low speed period. The receiver decoder works in the same manner as the normal IrDA mode.

### 25.3.11. Half-duplex communication mode

The half-duplex communication mode is enabled by setting the HDEN bit in USART_CTL2. The LMEN, CKEN bits in USART_CTL1 and SCEN, IREN bits in USART_CTL2 should be cleared in half-duplex communication mode.

Only one wire is used in half-duplex mode. The The TX and RX pins are connected together internally, and the RX pin is no longer used. The TX pin should be configured as open drain mode and communication conflicts are handled by software. When the TEN bit is set, the data in the data register will be sent.

### 25.3.12. Smartcard (ISO7816-3) mode

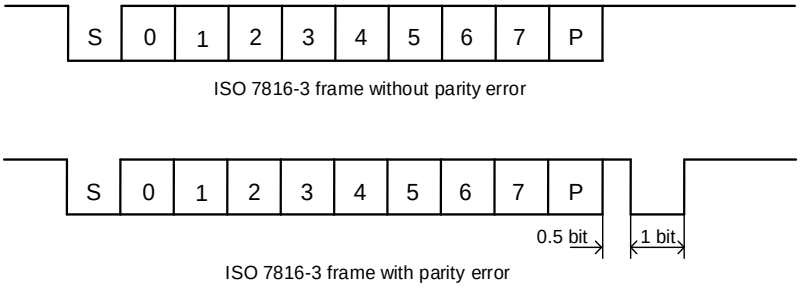
The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. Both the character (T = 0) mode and the block (T = 1) mode are supported. The smartcard mode is enabled by setting the SCEN bit in USART_CTL2. The LMEN bit in USART_CTL1 and HDEN, IREN bits in USART_CTL2 should be reset in smartcard mode.

A clock is provided to the smartcard if the CKEN bit is set. The clock can be divided for other use.

The frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits.

The smartcard mode is a half-duplex communication protocol. When connected to a smartcard, the TX pin must be configured as open drain mode, and drives a bidirectional line that is also driven by the smartcard.

**Figure 25-15. ISO7816-3 frame format**




#### Character (T = 0) mode

Compared to the timing in normal operation, the transmission time from transmit shift register to the TX pin is delayed by half baud clock, and the TC flag assertion time is delayed by a guard time that is configured by the GUAT[7:0] bits in USART_GP. In Smartcard mode, the internal guard time counter starts counting up after the stop bits of the last data frame, and the GUAT[7:0] bits should be configured as the character guard time (CGT) in ISO7816-3 protocol minus 12. The TC status is forced reset while the guard time counter is counting up. When the counter reaches the programmed value TC is asserted high.

During USART transmission, if a parity error event is detected, the smartcard may NACK the current frame by pulling down the TX pin during the last 1 bit time of the stop bits. The USART

can automatically resend data according to the protocol for SCRTNUM times. An interframe gap of 2 bits time will be inserted before the start of a resented frame. At the end of the last repeated character the TC bit is set immediately without guard time. The USART will stop transmitting and assert the frame error status if it still receives the NACK signal after the programmed number of retries. The USART will not take the NACK signal as the start bit.

During USART reception, if the parity error is detected in the current frame, the TX pin is pulled low during the last 1 bit time of the stop bits. This signal is the NACK signal to smartcard. Then a frame error occurs in smartcard side. The RBNE / receive DMA request is not activated if the received character is erroneous. According to the protocol, the smartcard can resend the data. The USART stops transmitting the NACK and the error is regarded as a parity error if the received character is still erroneous after the maximum number of retries which is specified in the SCRTNUM bit field. The NACK signal is enabled by setting the NKEN bit in USART_CTL2.

The idle frame and break frame are not supported in the Smartcard mode.

### Block (T = 1) mode

In block (T = 1) mode, the NKEN bit in the USART_CTL2 register should be cleared to deactivate the NACK transmission.

When requesting a read from the smartcard, the RT[23:0] bits in USART_RT register should be programmed with the BWT (block wait time) - 11 value and RBNEIE must be set. A timeout interrupt will be generated, if no answer is received from the card before the expiration of this period. If the first character is received before the expiration of the period, it is signaled by the RBNE interrupt. If DMA is used to read from the smartcard in block mode, the DMA must be enabled only after the first character is received.

In order to allow the automatic check of the maximum wait time between two consecutive characters, the USART_RT register must be programmed to the CWT (character wait time) - 11 value, which is expressed in baudtime units, after the reception of the first character (RBNE interrupt). The USART signals to the software through the RT flag and interrupt (when RTIE bit is set), if the smartcard doesn’t send a new character in less than the CWT period after the end of the previous character.

The USART uses a block length counter, which is reset when the USART is transmitting (TBE = 0), to count the number of received characters. The length of the block, which must be programmed in the BL[7:0] bits in the USART_RT register, is received from the smartcard in the third byte of the block (prologue field). This register field must be programmed to the minimum value (0x0), before the start of the block, when using DMA mode. With this value, an interrupt is generated after the 4th received character. The software must read the third byte as block length from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BL value. However, before the start of the block, the maximum value of BL (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

The total block length (including prologue, epilogue and information fields) equals BL+4. The end of the block is signaled to the software through the EBF flag and interrupt (when EBIE bit is set). The RT interrupt may occur in case of an error in the block length.

#### Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to H state of the line and parity is even. In this case, the following control bits must be programmed: MSBF = 0, DINV = 0.

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In this case, the following control bits must be programmed: MSBF = 1, DINV = 1.

### 25.3.13. ModBus communication

The USART offers basic support for the implementation of ModBus/RTU and ModBus/ASCII protocols by implementing an end of block detection.

In the ModBus/RTU mode, the end of one block is recognized by an idle line for more than 2 characters time. This function is implemented through the programmable timeout function.

To detect the idle line, the RTEN bit in the USART_CTL1 register and the RTIE in the USART_CTL0 register must be set. The USART_RT register must be set to the value corresponding to a timeout of 2 characters time. After the last stop bit is received, when the receive line is idle for this duration, an interrupt will be generated, informing the software that the current block reception is completed.

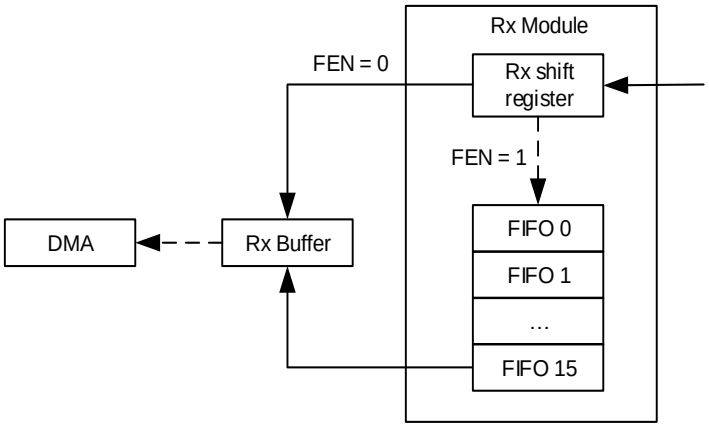
In the ModBus / ASCII mode, the end of a block is recognized by a specific (CR / LF) character sequence. The USART manages this mechanism using the character match function by programming the LF ASCII code in the ADDR0 or ADDR1 field and activating the address match interrupt (AMIE0 = 1 or AMIE1 = 1). When a LF has been received or can check the CR / LF in the DMA buffer, the software will be informed.

### 25.3.14. Receive / Transmitter FIFO

#### Receive FIFO

The receive FIFO can be enabled by setting the FEN bit of the USART_FCS register to avoid the overrun error when the CPU can’t serve the RBNE interrupt immediately. Up to 17 frames receive data can be stored in the receive FIFO and receive buffer. The RFF flag will be set when the receive FIFO is full, an interrupt is generated if the RFFIE bit is set.The RFT flag will be set when the receive FIFO reaches the threshold configured in the RFTCFG[2:0] bits., an interrupt is generated if the RFTIE bit is set. An interrupt will be generated when receive FIFO is not empty if RFNEIE bit is set.

**Figure 25-16. USART receive FIFO structure**

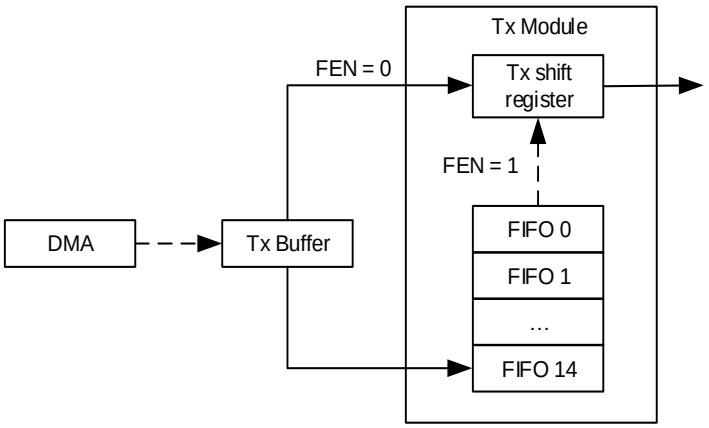



If the software read receive data buffer in the routing of the RBNE interrupt, the RBNEIE bit should be reset at the beginning of the routing and set after all of the receive data is read out. The PERR / NERR / FERR / EBF flags should be cleared before reading a receive data out.

### Transmit FIFO

The Transmitter FIFO can be enabled by setting the FEN bit of the USART_FCS register Up to 16 frames transmitter data can be stored in the transmitter FIFO and transmitter buffer. The TFE flag will be set when the transmitter FIFO is empty, an interrupt is generated if the TFEIE bit is set. The TFTIF flag will be set (or TFT flag will be clear) when the available space in transmitter FIFO reaches the threshold configured in the TFTCFG[2:0] bits, an interrupt is generated if the TFTIE bit is set. TFNF flag will be set when transmitter FIFO is not full, an interrupt will be generated if TFNFIE bit is set.

**Figure 25-17. USART transmitter FIFO structure**




## 25.3.15. Wakeup from deep-sleep mode

The USART is able to wake up the MCU from deep-sleep mode by the standard RBNE interrupt or the WUM interrupt.

The UESM bit must be set and the USART clock must be set to CK_IRC64MDIV or

CK_LXTAL (refer to the reset and clock unit RCU section).

When using the standard RBNE interrupt, the RBNEIE bit must be set before entering deep-sleep mode.

When using the WUIE interrupt, the source of WUIE interrupt may be selected through the WUM bit fields.

DMA must be disabled before entering deep-sleep mode. Before entering Deep-sleep mode, software must check that the USART is not performing a transfer, by checking the BSY flag in the USART_STAT register. The REA bit must be checked to ensure the USART is actually enabled.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUIE bit is set, independently of whether the MCU is in deep-Sleep or normal mode.

**Note:** AMEN0 or AMEN1 must be set when using address match method to wakeup MCU from Deep-Sleep mode.

### 25.3.16. USART interrupts

The USART interrupt events and flags are listed in ***Table 25-3. USART interrupt requests***.

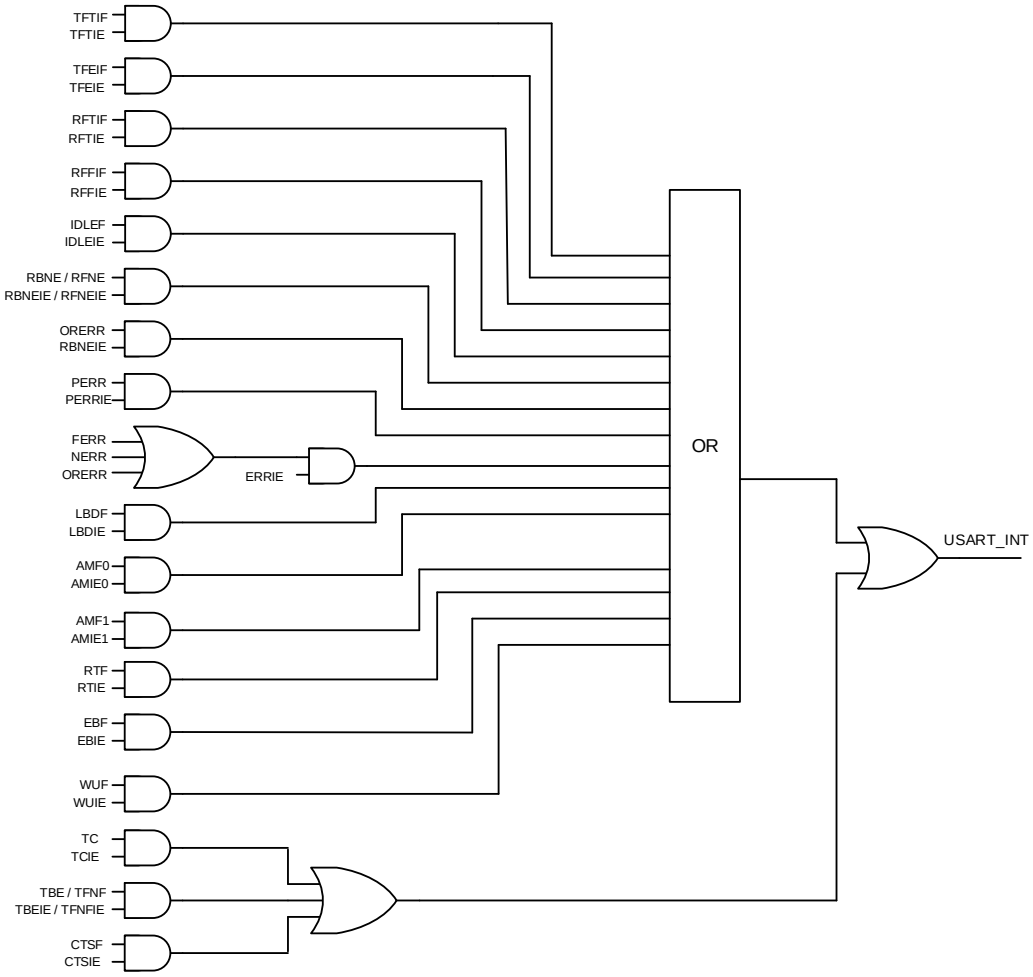
**Table 25-3. USART interrupt requests**

| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| Transmit data register empty or Transmit FIFO not full | TBE / TFNF | TBEIE / TFNFIE |
| CTS flag | CTSF | CTSIE |
| Transmission complete | TC | TCIE |
| Received data ready to be read or Receive FIFO not empty | RBNE/ RFNE | RBNEIE / RFNEIE |
| Overrun error detected | ORERR | |
| Receive FIFO full | RFFIF | RFFIE |
| Receive FIFO threshold reaches | RFT | RFTIE |
| Transmit FIFO empty | TFE | TFEIE |
| Transmit FIFO threshold reaches | TFTIF | TFTIE |
| Idle line detected | IDLEF | IDLEIE |
| Parity error flag | PERR | PERRIE |
| Break detected flag in LIN mode | LBDF | LBDIE |
| Reception errors (noise flag, overrun error, framing error) | NERR or ORERR or FERR | ERRIE |
| ADDR0 match | AMF0 | AMIE0 |

| Interrupt event | Event flag | Enable Control bit |
|---|---|---|
| ADDR1 match | AMF1 | AMIE1 |
| Receiver timeout error | RTF | RTIE |
| End of Block | EBF | EBIE |
| Wakeup from Deep-sleep mode | WUF | WUIE |

All of the interrupt events are ORed together before being sent to the interrupt controller, so the USART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine.

**Figure 25-18. USART interrupt mapping diagram**



## 25.4. Register definition

USART0 base address: 0x4001 1000

USART1 base address: 0x4000 4400

USART2 base address: 0x4000 4800

UART3 base address: 0x4000 4C00

UART4 base address: 0x4000 5000

USART5 base address: 0x4001 1400

UART6 base address: 0x4000 7800

UART7 base address: 0x4000 7C00

### 25.4.1. Control register 0 (USART_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AMIE1 | Reserved | | WL1 | EBIE | RTIE | DEA[4:0] | | | | | DED[4:0] | | | | |
| rw | | | rw | rw | rw | rw | | | | | rw | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OVSMOD | AMIE0 | MEN | WL0 | WM | PCEN | PM | PERRIE | TBEIE / TFNFIE | TCIE | RBNEIE / RFNEIE | IDLEIE | TEN | REN | UESM | UEN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | AMIE1 | ADDR1 character match interrupt enable<br>0: ADDR1 character match interrupt is disabled<br>1: ADDR1 character match interrupt is enabled |
| 30:29 | Reserved | Must be kept at reset value. |
| 28 | WL1 | Word length 1<br>This bit and WL0 bit determine the word length<br>00: 8 Data bits<br>01: 9 Data bits<br>10: 7 Data bits<br>11: 10 Data bits<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 27 | EBIE | End of block interrupt enable<br>0: End of block interrupt is disabled |

| | | |
|---|---|---|
| | | 1: End of block interrupt is enabled<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 26 | RTIE | Receiver timeout interrupt enable<br>0: Receiver timeout interrupt is disabled<br>1: Receiver timeout interrupt is enabled<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 25:21 | DEA[4:0] | Driver enable assertion time<br>These bits are used to define the time between the activation of the DE (driver enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time), which are configured by the OVSMOD bit.<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 20:16 | DED[4:0] | Driver enable de-assertion time<br>These bits are used to define the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (driver enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time), which are configured by the OVSMOD bit.<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 15 | OVSMOD | Oversample mode<br>0: Oversampling by 16<br>1: Oversampling by 8<br>This bit must be kept cleared in LIN, IrDA and smartcard modes.<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 14 | AMIE0 | ADDR0 character match interrupt enable<br>0: ADDR0 character match interrupt is disabled<br>1: ADDR0 character match interrupt is enabled |
| 13 | MEN | Mute mode enable<br>0: Mute mode disabled<br>1: Mute mode enabled |
| 12 | WL0 | Word length<br>This bit and WL1 bit determine the word length<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 11 | WM | Wakeup method in mute mode<br>0: Idle Line<br>1: Address match<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 10 | PCEN | Parity control enable<br>0: Parity control disabled<br>1: Parity control enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1). |

| | | |
|---|---|---|
| 9 | PM | Parity mode<br>0: Even parity<br>1: Odd parity<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 8 | PERRIE | Parity error interrupt enable<br>0: Parity error interrupt is disabled<br>1: An interrupt will occur whenever the PERR bit is set in USART_STAT |
| 7 | TBEIE | When FIFO is disabled:<br>Transmitter register empty interrupt enable<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the TBE bit is set in USART_STAT |
| | TFNFIE | When FIFO is enabled:<br>Transmit FIFO not full interrupt enable<br>0: Transmit FIFO not full interrupt is disabled<br>1: An interrupt will occur whenever the TFNF bit is set in USART_STAT |
| 6 | TCIE | Transmission complete interrupt enable<br>If this bit is set, an interrupt occurs when the TC bit in USART_STAT is set.<br>0: Transmission complete interrupt is disabled<br>1: Transmission complete interrupt is enabled |
| 5 | RBNEIE | When FIFO is disabled:<br>Read data buffer not empty interrupt and overrun error interrupt enable<br>0: Read data register not empty interrupt and overrun error interrupt disabled<br>1: An interrupt will occur whenever the ORERR bit is set or the RBNE bit is set in USART_STAT. |
| | RFNEIE | When FIFO is enabled:<br>Receive FIFO not empty interrupt and overrun error interrupt enable<br>0: Receive FIFO not empty interrupt and overrun error interrupt is disabled<br>1: An interrupt will occur whenever the ORERR bit is set or the RFNE bit is set in USART_STAT. |
| 4 | IDLEIE | IDLE line detected interrupt enable<br>0: IDLE line detected interrupt disabled<br>1: An interrupt will occur whenever the IDLEF bit is set in USART_STAT. |
| 3 | TEN | Transmitter enable<br>0: Transmitter is disabled<br>1: Transmitter is enabled |
| 2 | REN | Receiver enable<br>0: Receiver is disabled<br>1: Receiver is enabled and begins searching for a start bit |
| 1 | UESM | USART enable in Deep-sleep mode |

0: USART not able to wake up the MCU from Deep-sleep mode.
1: USART able to wake up the MCU from Deep-sleep mode. Providing that the clock source for the USART must be CK_IRC64MDIV or CK_LXTAL.
This bit is reserved in UART3 / UART4 / UART6 / UART7.

| | | |
|---|---|---|
| 0 | UEN | USART enable<br>0: USART prescaler and outputs disabled<br>1: USART prescaler and outputs enabled |

## 25.4.2. Control register 1 (USART_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR0[7:0] | | | | | | | | RTEN | Reserved | | | MSBF | DINV | TINV | RINV |
| rw | | | | | | | | rw | | | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STRP | LMEN | STB[1:0] | | CKEN | CPL | CPH | CLEN | Reserved | LBDIE | LBLEN | ADDM0 | Reserved | | | AMEN0 |
| rw | rw | rw | | rw | rw | rw | rw | | rw | rw | rw | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | ADDR0[7:0] | Address 0 of the USART terminal<br>These bits give the address 0 of the USART terminal.<br>In multiprocessor communication during mute mode or deep-sleep mode, this is used for wakeup with address match detection. The received frame, the MSB of which is equal to 1, will be compared to these bits. When the ADDM0 bit is reset, only the ADDR0[3:0] bits are used to compare.<br>In normal reception, these bits are also used for character detection. The whole received character (8-bit) is compared to the ADDR0[7:0] value and AMF0 flag is set on matching.<br>This bit field cannot be written when both reception (REN = 1) and USART (UEN = 1) are enabled. |
| 23 | RTEN | Receiver timeout enable<br>0: Receiver timeout function disabled<br>1: Receiver timeout function enabled<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 22:20 | Reserved | Must be kept at reset value. |
| 19 | MSBF | Most significant bit first<br>0: Data is transmitted / received with the LSB first<br>1: Data is transmitted / received with the MSB first<br>This bit field cannot be written when the USART is enabled (UEN = 1). |

| | | |
|---|---|---|
| 18 | DINV | Data bit level inversion<br>0: Data bit signal values are not inverted<br>1: Data bit signal values are inverted<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 17 | TINV | TX pin level inversion<br>0: TX pin signal values are not inverted<br>1: TX pin signal values are inverted<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 16 | RINV | RX pin level inversion<br>0: RX pin signal values are not inverted<br>1: RX pin signal values are inverted<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 15 | STRP | Swap TX / RX pins<br>0: The TX and RX pins functions are not swapped<br>1: The TX and RX pins functions are swapped<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 14 | LMEN | LIN mode enable<br>0: LIN mode disabled<br>1: LIN mode enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 13:12 | STB[1:0] | STOP bits length<br>00: 1 Stop bit<br>01: 0.5 Stop bits<br>10: 2 Stop bits<br>11: 1.5 Stop bits<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>**Note:** 0.5 and 1.5 stop bits are not used in UART3 / UART4 / UART6 / UART7. |
| 11 | CKEN | CK pin enable<br>0: CK pin disabled<br>1: CK pin enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 10 | CPL | Clock polarity<br>0: Steady low value on CK pin outside transmission window in synchronous mode<br>1: Steady high value on CK pin outside transmission window in synchronous mode<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 9 | CPH | Clock phase<br>0: The first clock transition is the first data capture edge in synchronous mode<br>1: The second clock transition is the first data capture edge in synchronous mode |

This bit field cannot be written when the USART is enabled (UEN = 1).

| | | |
|---|---|---|
| 8 | CLEN | CK length<br>0: The clock pulse of the last data bit (MSB) is not output to the CK pin in synchronous mode<br>1: The clock pulse of the last data bit (MSB) is output to the CK pin in synchronous mode<br>This bit field cannot be written when the USART is enabled (UEN = 1) |
| 7 | Reserved | Must be kept at reset value. |
| 6 | LBDIE | LIN break detection interrupt enable<br>0: LIN break detection interrupt is disabled<br>1: An interrupt will occur whenever the LBDF bit is set in USART_STAT<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 5 | LBLEN | LIN break frame length<br>0: 10 bit break detection<br>1: 11 bit break detection<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 4 | ADDM0 | Address 0 detection mode<br>This bit is used to select between 4-bit address detection and full-bit address 0 detection.<br>0: 4-bit address detection<br>1: Full-bit address detection. In 7-bit, 8-bit and 9-bit data modes, the address detection is done on 6-bit, 7-bit and 8-bit address (ADDR0[5:0], ADDR0[6:0] and ADDR0[7:0]) respectively<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 3:1 | Reserved | Must be kept at reset value. |
| 0 | AMEN0 | Address 0 match mode enable<br>0: Address 0 match mode disable<br>1: Address 0 match mode enable. |

## 25.4.3. Control register 2 (USART_CTL2)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR1[7:0] | | | | | | | | ADDM1 | WUIE | WUM[1:0] | | SCRTNUM[2:0] | | | AMEN1 |
| rw | | | | | | | | rw | rw | rw | | rw | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DEP | DEM | DDRE | OVRD | OSB | CTSIE | CTSEN | RTSEN | DENT | DENR | SCEN | NKEN | HDEN | IRLP | IREN | ERRIE |

rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | ADDR1[7:0] | Address 1 of the USART terminal<br>These bits give the address 1 of the USART terminal.<br>In multiprocessor communication during mute mode or Deep-sleep mode, this is used for wakeup with address mark detection. The received frame, the MSB of which is equal to 1, will be compared to these bits. When the ADDM1 bit is reset, only the ADDR1 [3:0] bits are used to compare.<br>In normal reception, these bits are also used for character detection. The whole received character (8-bit) is compared to the ADDR1[7:0] value and AMF1 flag is set on matching.<br>This bit field cannot be written when both reception (REN = 1) and USART are enabled (UEN = 1). |
| 23 | ADDM1 | Address 1 detection mode<br>This bit is used to select between 4-bit address detection and full-bit address 1 detection.<br>0: 4-bit address detection<br>1: Full-bit address detection. In 7-bit, 8-bit and 9-bit data modes, the address detection is done on 6-bit, 7-bit and 8-bit address (ADDR1[5:0], ADDR1[6:0] and ADDR1[7:0]) respectively.<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 22 | WUIE | Wakeup from deep-sleep mode interrupt enable<br>0: Wakeup from deep-sleep mode interrupt is disabled<br>1: Wakeup from deep-sleep mode interrupt is enabled<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 21:20 | WUM[1:0] | Wakeup mode from deep-sleep mode<br>These bits are used to specify the event which activates the WUF (wakeup from deep-sleep mode flag) in the USART_STAT register.<br>00: WUF active on address match, which is defined by ADDR0 and ADDM0 or ADDR1 and ADDM1.<br>01: Reserved<br>10: WUF active on start bit<br>11: WUF active on RBNE<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 19:17 | SCRTNUM[2:0] | Smartcard auto-retry number<br>In smartcard mode, these bits specify the number of retries in transmission and reception.<br>In transmission mode, a transmission error (FERR bit set) will occur after this number of automatic retransmission retries.<br>In reception mode, reception error (RBNE and PERR bits set) will occur after this |

| | | |
|---|---|---|
| | | number or erroneous reception trials.<br>When these bits are configured as 0x0, there will be no automatic retransmission in transmit mode.<br>This bit field is only can be cleared to 0 when the USART is enabled (UEN = 1), to stop retransmission.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 16 | AMEN1 | Address 1 match mode enable<br>0: Address 1 match mode disable<br>1: Address 1 match mode enable |
| 15 | DEP | Driver enable polarity mode<br>0: DE signal is active high<br>1: DE signal is active low<br>This bit field cannot be written when the USART is enabled (UEN = 1) |
| 14 | DEM | Driver enable mode<br>This bit is used to activate the external transceiver control, through the DE signal, which is output on the RTS pin.<br>0: DE function is disabled<br>1: DE function is enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 13 | DDRE | Mask DMA request on reception error.<br>0: DMA is not disabled in case of reception error. The DMA request is not asserted to make sure the erroneous data is not transferred, but the next correct received data will be transferred. The RBNE is kept 0 to prevent overrun when reception error, but the corresponding error flag is set. This mode can be used in smartcard mode<br>1: The DMA request is not asserted in case of reception error until the error flag is cleared. The RBNE flag and corresponding error flag will be set. The software must first disable the DMA request (DENR = 0) or clear RBNE / RFNE in USART_STAT when FIFO mode is enabled before clearing the error flag<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 12 | OVRD | Overrun disable<br>0: Overrun functionality is enabled. The ORERR error flag will be set when received data is not read before receiving new data, and the new data will be lost<br>1: Overrun functionality is disabled. The ORERR error flag will not be set when received data is not read before receiving new data, and the new received data overwrites the previous content of the USART_RDATA register. When FIFO mode is enabled, the data is written in USART_RDATA directly and Receive FIFO is bypassed. Even if FIFO is enabled, the RBNE bit is to be used.<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 11 | OSB | One sample bit method<br>0: Three sample bit method |

| | | |
|---|---|---|
| | | 1: One sample bit method<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 10 | CTSIE | CTS interrupt enable<br>0: CTS interrupt is disabled<br>1: An interrupt will occur whenever the CTS bit is set in USART_STAT |
| 9 | CTSEN | CTS enable<br>0: CTS hardware flow control disabled<br>1: CTS hardware flow control enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 8 | RTSEN | RTS enable<br>0: RTS hardware flow control disabled<br>1: RTS hardware flow control enabled, data can be requested only when there is space in the receive buffer<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 7 | DENT | DMA enable for transmission<br>0: DMA mode is disabled for transmission<br>1: DMA mode is enabled for transmission |
| 6 | DENR | DMA enable for reception<br>0: DMA mode is disabled for reception<br>1: DMA mode is enabled for reception |
| 5 | SCEN | Smartcard mode enable<br>0: Smartcard mode disabled<br>1: Smartcard mode enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 4 | NKEN | NACK enable in Smartcard mode<br>0: Disable NACK transmission when parity error<br>1: Enable NACK transmission when parity error<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 3 | HDEN | Half-duplex enable<br>0: Half duplex mode is disabled<br>1: Half duplex mode is enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 2 | IRLP | IrDA low-power<br>0: Normal mode<br>1: Low-power mode<br>This bit field cannot be written when the USART is enabled (UEN = 1). |
| 1 | IREN | IrDA mode enable |

| | | |
|---|---|---|
| | | 0: IrDA disabled<br>1: IrDA enabled<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 0 | ERRIE | Error interrupt enable<br>0: Error interrupt disabled<br>1: An interrupt will occur whenever the FERR bit or the ORERR bit or the NERR bit is set in USART_STAT in multibuffer communication |

### 25.4.4. Baud rate generator register (USART_BAUD)

Address offset: 0x0C
Reset value: 0x0000 0000

This register cannot be written when the USART is enabled (UEN=1).

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRR [15:4] | | | | | | | | | | | | BRR[3:0] | | | |
| rw | | | | | | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:4 | BRR[15:4] | Integer of baud-rate divider<br>INTDIV = BRR[15:4] |
| 3:0 | BRR [3:0] | Fraction of baud-rate divider<br>If OVSMOD = 0, FRADIV = BRR [3:0];<br>If OVSMOD = 1, FRADIV = BRR [2:0], BRR [3] must be reset. |

### 25.4.5. Prescaler and guard time configuration register (USART_GP)

Address offset: 0x10
Reset value: 0x0000 0000

This register cannot be written when the USART is enabled (UEN = 1).

This register is reserved in UART3 / UART4 / UART6 / UART7.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GUAT[7:0] | | | | | | | | PSC[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:8 | GUAT[7:0] | Guard time value in smartcard mode<br>This bit field cannot be written when the USART is enabled (UEN=1). |
| 7:0 | PSC[7:0] | Prescaler value for dividing the system clock<br>In IrDA Low-power mode, the division factor is the prescaler value.<br>00000000: Reserved - do not program this value<br>00000001: divides the source clock by 1<br>00000010: divides the source clock by 2<br>...<br>In IrDA normal mode,<br>00000001: can be set this value only<br>In smartcard mode, the prescaler value for dividing the system clock is stored in PSC[4:0] bits. And the bits of PSC[7:5] must be kept at reset value. The division factor is twice as the prescaler value.<br>00000: Reserved - do not program this value<br>00001: divides the source clock by 2<br>00010: divides the source clock by 4<br>00011: divides the source clock by 6<br>...<br>This bit field cannot be written when the USART is enabled (UEN=1). |

## 25.4.6. Receiver timeout register (USART_RT)

Address offset: 0x14

Reset value: 0x0000 0000

This bit is reserved in UART3 / UART4 / UART6 / UART7.

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BL[7:0] | | | | | | | | RT[23:16] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:24 | BL[7:0] | Block Length<br>These bits specify the block length in smartcard T=1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC / 2-CRC) - 1.<br>This value, which must be programmed only once per received block, can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). The block length counter is reset when TBE = 0 or TFE = 0 ( FIFO is enabled) in smartcard mode.<br>In other modes, when REN = 0 (receiver disabled) and/or when the EBC bit is written to 1, the block length counter is reset. |
| 23:0 | RT[23:0] | Receiver timeout threshold<br>These bits are used to specify receiver timeout value in terms of number of baud clocks.<br>In standard mode, the RTF flag is set if no new start bit is detected for more than the RT value after the last received character.<br>In smartcard mode, the CWT and BWT are implemented by this value. In this case, the timeout measurement is started from the start bit of the last received character.<br>These bits can be written on the fly. The RTF flag will be set if the new value is lower than or equal to the counter. These bits must only be programmed once per received character.<br>**Note：** When oversampling by 16, RT[23:0] cannot be configured as 0xFFFFFF. |

## 25.4.7. Command register (USART_CMD)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | TXFCMD | RXFCMD | MMCMD | SBKCMD | Reserved |
| | | | | | | | | | | | w | w | w | w | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | TXFCMD | Transmit data flush request<br>When FIFO is disabled, Writing 1 to this bit sets the TBE flag, to discard the transmit data.<br>When FIFO is enabled, Writing 1 to this bit flushes the whole Transmit FIFO and sets TFE flag in USART_FCS.<br>Note:When FIFO is enabled, druing the flush command,TFNF is reset until Transmit |

| | | |
|---|---|---|
| | | FIFO is empty. |
| 3 | RXFCMD | Receive data flush command<br>When FIFO is disabled, Writing 1 to this bit clears the RBNE flag to discard the received data without reading it.<br>When FIFO is enabled, Writing 1 to this bit empties the Receive FIFO and sets RFNE flag in USART_FCS. |
| 2 | MMCMD | Mute mode command<br>Writing 1 to this bit makes the USART into mute mode and sets the RWU flag. |
| 1 | SBKCMD | Send break command<br>Writing 1 to this bit sets the SBF flag and makes the USART send a BREAK frame, as soon as the transmit machine is idle. |
| 0 | Reserved | Must be kept at reset value. |

### 25.4.8. Status register (USART_STAT)

Address offset: 0x1C

Reset value: 0x0000 00C0

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | REA | TEA | WUF | RWU | SBF | AMF0 | BSY |
| | | | | | | | | | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | AMF1 | EBF | RTF | CTS | CTSF | LBDF | TBE<br>TFNF | TC | RBNE<br>RFNE | IDLEF | ORERR | NERR | FERR | PERR |
| | | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | REA | Receive enable acknowledge flag<br>This bit, which is set / reset by hardware, reflects the receive enable state of the USART core logic.<br>0: The USART core receiving logic has not been enabled<br>1: The USART core receiving logic has been enabled<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 21 | TEA | Transmit enable acknowledge flag<br>This bit, which is set / reset by hardware, reflects the transmit enable state of the USART core logic.<br>0: The USART core transmitting logic has not been enabled<br>1: The USART core transmitting logic has been enabled |
| 20 | WUF | Wakeup from deep-sleep mode flag |

| | | |
|---|---|---|
| | | 0: No wakeup from deep-sleep mode<br>1: Wakeup from deep-sleep mode. An interrupt is generated if WUFIE = 1 in the USART_CTL2 register and the MCU is in Deep-sleep mode.<br>This bit is set by hardware when a wakeup event, which is defined by the WUM bit field, is detected.<br>Cleared by writing a 1 to the WUC in the USART_INTC register.<br>This bit can also be cleared when UESM is cleared.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 19 | RWU | Receiver wakeup from mute mode<br>This bit is used to indicate if the USART is in mute mode.<br>0: Receiver in active mode<br>1: Receiver in mute mode<br>It is cleared/set by hardware when a wakeup/mute sequence (address or IDLEIE) is recognized, which is selected by the WM bit in the USART_CTL0 register.<br>This bit can only be set by writing 1 to the MMCMD bit in the USART_CMD register when wakeup on IDLEIE mode is selected. |
| 18 | SBF | Send break flag<br>0: No break character is transmitted<br>1: Break character will be transmitted<br>This bit indicates that a send break character was requested.<br>Set by software, by writing 1 to the SBKCMD bit in the USART_CMD register.<br>Cleared by hardware during the stop bit of break transmission. |
| 17 | AMF0 | ADDR0 character match flag<br>0: ADDR0 character does not match the received character<br>1: ADDR0 character matches the received character, an interrupt is generated if AMIE0=1 in the USART_CTL0 register.<br>Set by hardware, when the character defined by ADDR0[7:0] is received.<br>Cleared by writing 1 to the AMC in the USART_INTC register. |
| 16 | BSY | Busy flag<br>0: USART reception path is idle<br>1: USART reception path is working |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | AMF1 | ADDR1 character match flag<br>0: ADDR1 character does not match the received character<br>1: ADDR1 character matches the received character, an interrupt is generated if AMIE1 = 1 in the USART_CTL0 register.<br>Set by hardware, when the character defined by ADDR1[7:0] is received.<br>Cleared by writing 1 to the AMC in the USART_INTC register. |
| 12 | EBF | End of block flag<br>0: End of Block not reached<br>1: End of Block (number of characters) reached. An interrupt is generated if the |

| | | |
|---|---|---|
| | | EBIE = 1 in the USART_CTL1 register<br>Set by hardware when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.<br>Cleared by writing 1 to EBC bit in USART_INTC register.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 11 | RTF | Receiver timeout flag<br>0: Timeout value not reached<br>1: Timeout value reached without any data reception. An interrupt is generated if RTIE bit in the USART_CTL1 register is set.<br>Set by hardware when the RT value, programmed in the USART_RT register has lapsed without any communication.<br>Cleared by writing 1 to RTC bit in USART_INTC register.<br>The timeout corresponds to the CWT or BWT timings in smartcard mode.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 10 | CTS | CTS level<br>This bit equals to the inverted level of the nCTS input pin.<br>0: nCTS input pin is in high level<br>1: nCTS input pin is in low level |
| 9 | CTSF | CTS change flag<br>0: No change occurred on the nCTS status line<br>1: A change occurred on the nCTS status line. An interrupt will occur if the CTSIE bit is set in USART_CTL2<br>Set by hardware when the nCTS input toggles.<br>Cleared by writing 1 to CTSC bit in USART_INTC register. |
| 8 | LBDF | LIN break detected flag<br>0: LIN Break is not detected<br>1: LIN Break is detected. An interrupt will occur if the LBDIE bit is set in USART_CTL1<br>Set by hardware when the LIN break is detected.<br>Cleared by writing 1 to LBDC bit in USART_INTC register.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 7 | TBE | When FIFO is disabled:<br>Transmit data register empty<br>0: Data is not transferred to the shift register<br>1: Data is transferred to the shift register. An interrupt will occur if the TBEIE bit is set in USART_CTL0<br>Set by hardware when the content of the USART_TDATA register has been transferred into the transmit shift register or writing 1 to TXFCMD bit of the USART_CMD register.<br>Cleared by a write to the USART_TDATA. |
| | TFNF | When FIFO is enabled: |

| | | |
|---|---|---|
| | | Transmit FIFO not full<br>0: Transmit FIFO is full<br>1: Transmit FIFO is not full. An interrupt will occur if the TFNFIE bit is set in USART_CTL0.<br>Set by hardware when the transmit FIFO is not full. The flag is cleared when the transmit FIFO is full.<br>**Note:** The TFNF bit keep reset during the TXFCMD set until Transmit FIFO is empty. |
| 6 | TC | Transmission completed<br>0: Transmission is not completed<br>1: Transmission is complete. An interrupt will occur if the TCIE bit is set in USART_CTL0.<br>Set by hardware if the transmission of a frame containing data is completed and if the TBE or TFE bit in USART_FCS is set.<br>Cleared by writing 1 to TCC bit in USART_INTC register.<br>Note: The TC bit is set immediately when TEN is reset and no transmission is on going. |
| 5 | RBNE | When FIFO is disabled:<br>Read data buffer not empty<br>0: Data is not received<br>1: Data is received and ready to be read. An interrupt will occur if the RBNEIE bit is set in USART_CTL0.<br>Set by hardware when the content of the receive shift register has been transferred to the USART_RDATA.<br>Cleared by reading the USART_RDATA or writing 1 to RXFCMD bit of the USART_CMD register. |
| | RFNE | When FIFO is enabled:<br>Receive FIFO not empty<br>0: Receive FIFO is empty<br>1: Receive FIFO is not empty. An interrupt will occur if the RFNEIE bit is set in USART_CTL0.<br>Set by hardware when the receive FIFO is not empty. The flag is cleared when the receive FIFO is empty. The bit can also be reset by setting RXFCMD bit in USART_CMD. |
| 4 | IDLEF | IDLE line detected flag<br>0: No Idle line is detected<br>1: Idle line is detected. An interrupt will occur if the IDLEIE bit is set in USART_CTL0<br>Set by hardware when an Idle line is detected. It will not be set again until the RBNE bit or RFNE bit has been set itself.<br>Cleared by writing 1 to IDLEC bit in USART_INTC register. |
| 3 | ORERR | Overrun error<br>0: No overrun error is detected<br>1: Overrun error is detected. An interrupt will occur if the RBNEIE or RFNEIE bit is |

| | | |
|---|---|---|
| | | set in USART_CTL0. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.<br>Set by hardware when the word in the receive shift register is ready to be transferred into the USART_RDATA register while the RBNE bit or RFF bit in USART_FCS is set.<br>Cleared by writing 1 to OREC bit in USART_INTC register. |
| 2 | NERR | Noise error flag<br>0: No noise error is detected<br>1: Noise error is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.<br>Set by hardware when noise error is detected on a received frame.<br>Cleared by writing 1 to NEC bit in USART_INTC register.<br>Note: When this bit and RBNE or RFNE bit appears at the same time, it does not generate an interrupt. When FIFO is enabled, the error is associated with the data in the USART_RDATA. |
| 1 | FERR | Frame error flag<br>0: No framing error is detected<br>1: Frame error flag or break character is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.<br>Set by hardware when a de-synchronization, excessive noise or a break character is detected. This bit will be set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame), when USART transmits in smartcard mode.<br>Cleared by writing 1 to FEC bit in USART_INTC register.<br>Note: When FIFO is enabled, the error is associated with the data in the USART_RDATA. |
| 0 | PERR | Parity error flag<br>0: No parity error is detected<br>1: Parity error flag is detected. An interrupt will occur if the PERRIE bit is set in USART_CTL0.<br>Set by hardware when a parity error occurs in receiver mode.<br>Cleared by writing 1 to PEC bit in USART_INTC register.<br>Note: When FIFO is enabled, the error is associated with the data in the USART_RDATA. |

### 25.4.9. Interrupt status clear register (USART_INTC)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | WUC | Reserved | | AMC0 | AMC1 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | w | | | w | w |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | EBC | RTC | Reserved | CTSC | LBDC | Reserved | TCC | Reserved | IDLEC | OREC | NEC | FEC | PEC |
| | | | w | w | | w | w | | w | | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | WUC | Wakeup from deep-sleep mode clear<br>Writing 1 to this bit clears the WUF bit in the USART_STAT register.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 19:18 | Reserved | Must be kept at reset value. |
| 17 | AMC0 | ADDR0 character match clear<br>Writing 1 to this bit clears the AMF0 bit in the USART_STAT register. |
| 16 | AMC1 | ADDR1 character match clear<br>Writing 1 to this bit clears the AMF1 bit in the USART_STAT register. |
| 15:13 | Reserved | Must be kept at reset value. |
| 12 | EBC | End of block clear<br>Writing 1 to this bit clears the EBF bit in the USART_STAT register.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 11 | RTC | Receiver timeout clear<br>Writing 1 to this bit clears the RTF flag in the USART_STAT register.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 10 | Reserved | Must be kept at reset value. |
| 9 | CTSC | CTS change clear<br>Writing 1 to this bit clears the CTSF bit in the USART_STAT register. |
| 8 | LBDC | LIN break detected clear<br>Writing 1 to this bit clears the LBDF flag in the USART_STAT register.<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |
| 7 | Reserved | Must be kept at reset value. |
| 6 | TCC | Transmission complete clear<br>Writing 1 to this bit clears the TC bit in the USART_STAT register. |
| 5 | Reserved | Must be kept at reset value. |
| 4 | IDLEC | Idle line detected clear<br>Writing 1 to this bit clears the IDLEF bit in the USART_STAT register. |
| 3 | OREC | Overrun error clear<br>Writing 1 to this bit clears the ORERR bit in the USART_STAT register. |

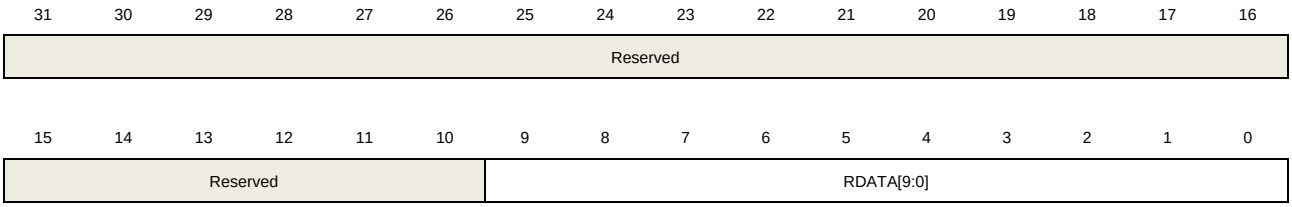
| | | |
|---|---|---|
| 2 | NEC | Noise detected clear<br>Writing 1 to this bit clears the NERR bit in the USART_STAT register. |
| 1 | FEC | Frame error flag clear<br>Writing 1 to this bit clears the FERR bit in the USART_STAT register |
| 0 | PEC | Parity error clear<br>Writing 1 to this bit clears the PERR bit in the USART_STAT register. |

### 25.4.10. Receive data register (USART_RDATA)

Address offset: 0x24

Reset value: Undefined

This register has to be accessed by word (32-bit)



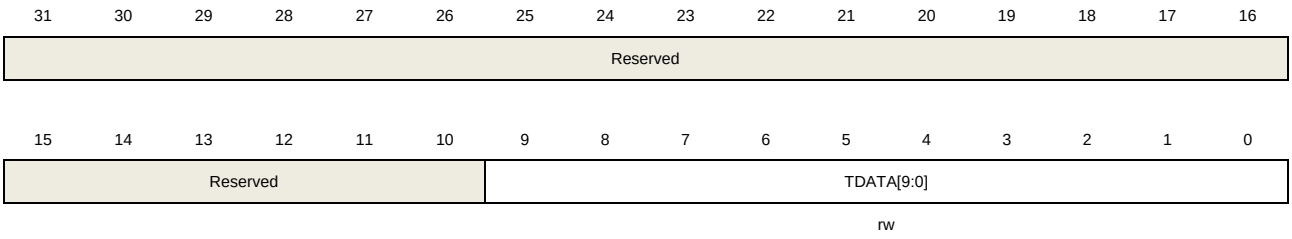

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9:0 | RDATA[9:0] | Receive data value<br>The received data character is contained in these bits.<br>The value read in the MSB (bit6, bit7, bit8 or bit9 depending on the data length) will be the received parity bit, if receiving with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register). |

### 25.4.11. Transmit data register (USART_TDATA)

Address offset: 0x28

Reset value: Undefined

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value |
| 9:0 | TDATA[9:0] | Transmit Data value<br>The transmit data character is contained in these bits.<br>The value written in the MSB (bit6, bit7, bit8 or bit9 depending on the data length) will be replaced by the parity, when transmitting with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register).<br>This register must be written only when TBE bit in USART_STAT register is set. |

## 25.4.12. USART coherence control register (USART_CHC)

Address offset: 0xC0

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | EPERR | Reserved | | | | | | | HCM |
| | | | | | | | rc_w0 | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | EPERR | Early parity error flag. This flag will be set as soon as the parity bit has been detected, which is before RBNE flag. This flag is cleared by writing 0.<br>0: No parity error is detected<br>1: Parity error is detected. |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | HCM | Hardware flow control coherence mode<br>0: nRTS signal equals to the RBNE in status register<br>1: nRTS signal is set when the last data bit (parity bit when pce is set) has been sampled. |

## 25.4.13. USART FIFO control and status register (USART_FCS)

Address offset: 0xD0

Reset value: 0x0300 0400

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TFEIE | Reserved | TFTIE | Reserved | RFTIE | TFEC | TFTIF | TFEIF | Reserved | RFTIF | TFTCFG[2:0] | | | RFTCFG[2:0] | | |
| rw | | rw | | rw | rw | r | r | | rc_w0 | rw | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFFIF | RFCNT[2:0] | | | RFF | RFE | RFFIE | FEN | TFF | TFE | TFT | RFT | Reserved | RFCNT[4:3] | | ELNACK |
| rc_w0 | r | | | r | r | rw | rw | r | r | r | r | | r | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TFEIE | Transmit FIFO empty interrupt enable<br>If this bit is set, an interrupt occurs whenever the TFE bit is set.<br>0: Transmit FIFO empty interrupt is disabled<br>1: Transmit FIFO empty interrupt is enabled |
| 30 | Reserved | Must be kept at reset value. |
| 29 | TFTIE | Transmit FIFO threshold interrupt enable<br>If this bit is set, an interrupt will occur whenever the avaliable space in transmit FIFO reached the threshold configured in TFTCFG[2:0].<br>0: Transmit FIFO threshold interrupt is disabled<br>1: Transmit FIFO threshold interrupt is enabled |
| 28 | Reserved | Must be kept at reset value. |
| 27 | RFTIE | Receive FIFO threshold interrupt enable<br>If this bit is set, an interrupt will occur whenever the receive FIFO reached the threshold configured in RFTCFG.<br>0: Receive FIFO threshold interrupt is disabled<br>1: Receive FIFO threshold interrupt is enabled |
| 26 | TFEC | Transmit FIFO empty flag clear<br>Writing 1 to this bit clears the TFE flag. |
| 25 | TFTIF | Transmit FIFO threshold interrupt flag<br>The bit is valid when TFTIE bit is set.<br>0: The available space in transmit FIFO does not reach the programmed threshold<br>1: The available space in transmit FIFO reached the programmed threshold. An interrupt will occur if the TFTIE bit is set.<br>Set by hardware when the available space in transmit FIFO reaches the threshold configured in TFTCFG[2:0]. |
| 24 | TFEIF | Transmit FIFO empty interrupt flag<br>The bit is valid when TFEIE bit is set.<br>0: Transmit FIFO is not empty<br>1: Transmit FIFO is empty |
| 23 | Reserved | Must be kept at reset value. |
| 22 | RFTIF | Receive FIFO threshold interrupt flag<br>The bit is valid when RFTIE bit is set.<br>0: Receive FIFO does not reach the programmed threshold<br>1: Receive FIFO reached the programmed threshold |

| | | |
|---|---|---|
| 21:19 | TFTCFG[2:0] | Transmit FIFO threshold configuration<br>000:Transmit FIFO reaches 1/8 of its depth<br>001:Transmit FIFO reaches 1/4 of its depth<br>010:Transmit FIFO reaches 1/2 of its depth<br>011:Transmit FIFO reaches 3/4 of its depth<br>100:Transmit FIFO reaches 7/8 of its depth<br>101:Transmit FIFO becomes empty<br>11x: Reserved |
| 18:16 | RFTCFG[2:0] | Receive FIFO threshold configuration<br>000:Receive FIFO reaches 1/8 of its depth<br>001:Receive FIFO reaches 1/4 of its depth<br>010:Receive FIFO reaches 1/2 of its depth<br>011:Receive FIFO reaches 3/4 of its depth<br>100:Receive FIFO reaches 7/8 of its depth<br>101:Receive FIFO becomes full<br>11x: Reserved |
| 15 | RFFIF | Receive FIFO full interrupt flag<br>The bit is valid when RFFIE bit is set.<br>0: Receive FIFO is not full<br>1: Receive FIFO is full |
| 14:12 | RFCNT[2:0] | Receive FIFO counter number<br>These bits and RFCNT[4:3] bits determine the receive FIFO counter number. |
| 11 | RFF | Receive FIFO full flag<br>0: Receive FIFO is not full<br>1: Receive FIFO is full. An interrupt will occur if the RFFIE bit is set.<br>Set by hardware when the number of received data is RXFIFO size + 1. |
| 10 | RFE | Receive FIFO empty flag<br>0: Receive FIFO is not empty<br>1: Receive FIFO is empty. |
| 9 | RFFIE | Receive FIFO full interrupt enable<br>If this bit is set, an interrupt occurs when the RFF bit is set.<br>0: Receive FIFO full interrupt is disable<br>1: Receive FIFO full interrupt is enable |
| 8 | FEN | FIFO enable<br>0: FIFO is disable<br>1: FIFO is enable<br>This bit field cannot be written when the USART is enabled (UEN = 1).<br>Note: Do not change the FEN bit when receiving or transmitting data is not accomplished. When UEN is cleared and reconfigure the UEN without changing the FEN bit, please flush the FIFO if do not need the pervious FIFO value. |

| | | |
|---|---|---|
| 7 | TFF | Transmit FIFO full flag<br>0: Transmit FIFO is not full<br>1: Transmit FIFO is full |
| 6 | TFE | Transmit FIFO empty flag<br>0: Transmit FIFO is not empty<br>1: Transmit FIFO is empty. An interrupt will occur if the TFEIE bit is set.<br>Set by hardware when the transmit FIFO is empty. The flag is cleared when the transmit FIFO has at least one data. The bit can also be set by setting TXFCMD bit in USART_CMD. |
| 5 | TFT | Transmit FIFO threshold flag<br>0: The available space in transmit FIFO reached the programmed threshold.<br>1: The available space in transmit FIFO does not reach the programmed threshold<br>Reset by hardware when the available space in transmit FIFO reaches the threshold configured in TFTCFG[2:0]. |
| 4 | RFT | Receive FIFO threshold flag<br>0: Receive FIFO does not reach the programmed threshold<br>1: Receive FIFO reached the programmed threshold. An interrupt will occur if the RFTIE bit is set.<br>Set by hardware when the receive FIFO reaches the threshold configured in RFTCFG[2:0]. This means that there are (RFTCFG[2:0] - 1) data in the Receive FIFO and one data in the USART_RDATA register<br>**Note:** When the RFTCFG[2:0] = 0b101 and 16 data are available, RFT flag will be set. |
| 3 | Reserved | Must be kept at reset value. |
| 2:1 | RFCNT[4:3] | Receive FIFO counter number<br>These bits and RFCNT[2:0] bits determine the receive FIFO counter number. |
| 0 | ELNACK | Early NACK when smartcard mode is selected.<br>The NACK pulse occurs 1/16 bit time earlier when the parity error is detected.<br>0:Early NACK disable when smartcard mode is selected<br>1:Early NACK enable when smartcard mode is selected<br>This bit is reserved in UART3 / UART4 / UART6 / UART7. |

# 26. Inter-integrated circuit interface (I2C)

## 26.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard mode, fast mode and fast mode plus as well as CRC calculation and checking, SMBus (system management bus), and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

## 26.2. Characteristics

- Parallel-bus to I2C-bus protocol converter and interface.
- Both master and slave functions with the same interface.
- Bi-directional data transfer between master and slave.
- Supports 7-bit and 10-bit addressing and general call addressing.
- Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask).
- Programmable setup time and hold time.
- Multi-master capability.
- Supports standard mode (up to 100 kHz) and fast mode (up to 400 kHz) and fast mode plus (up to 1MHz, this mode must be enabled in SYSCFG_PMCFG).
- Configurable SCL stretching in slave mode.
- Supports DMA mode.
- SMBus 3.0 and PMBus 1.3 compatible.
- Optional PEC (packet error checking) generation and check.
- Programmable analog and digital noise filters.
- Wakeup from sleep mode and Deep-sleep mode on I2C address match.
- Independent clock from PCLK.

## 26.3. Function overview

*Figure 26-1. I2C module block diagram* below provides details on the internal configuration of the I2C interface.

**Figure 26-1. I2C module block diagram**

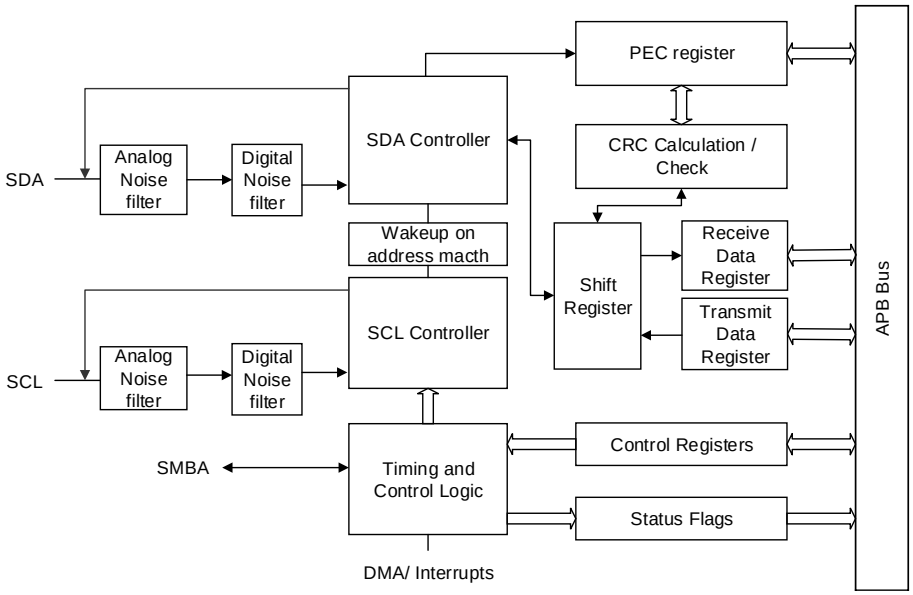



**Table 26-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)**

| Term | Description |
|---|---|
| Transmitter | the device which sends data to the bus |
| Receiver | the device which receives data from the bus |
| Master | the device which initiates a transfer, generates clock signals and terminates a transfer |
| Slave | the device addressed by a master |
| Multi-master | more than one master can attempt to control the bus at the same time without corrupting the message |
| Arbitration | procedure to ensure that, if more than one master tries to control the bus simultaneously, only one is allowed to do so and the winning master's message is not corrupted |

### 26.3.1. Clock requirements

The I2C clock is independent of the PCLK frequency, so that the I2C can be operated independently.

This I2C clock (I2CCLK) can be selected from the following three clock sources:

- PCLK1: APB1 clock (default value)
- PLL2R: Phase Lock Loop
- IRC64M: Internal 64M RC oscillator
- LPIRC4M: Low Power Internal 4M RC oscillator

The I2CCLK period $t_{I2CCLK}$ must match the conditions as follows:

- $t_{I2CCLK} < (t_{LOW} - t_{filters})/4$

- $t_{I2CCLK} < t_{HIGH}$

with:

$t_{LOW}$: SCL low time

$t_{HIGH}$: SCL high time

$t_{filters}$: When the filters are enabled, represent the delays by the analog filter and digital filter.

Analog filter delay is maximum 130ns. Digital filter delay is DNF[3:0]×$t_{I2CCLK}$.

The period of PCLK clock $t_{PCLK}$ match the conditions as follows:

- $t_{PCLK} < 4/3 * t_{SCL}$

with:

$t_{SCL}$: the period of SCL

**Note:** When the I2C kernel is provided by PCLK, this clock must match the conditions for $t_{I2CCLK}$.

### 26.3.2. I2C communication flow

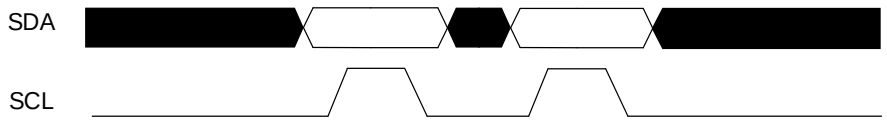
An I2C device is able to transmit or receive data whether it's a master or a slave, thus, there're 4 operation modes for an I2C device:

- Slave transmitter
- Slave receiver
- Master transmitter
- Master receiver

#### Data validation

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see ***Figure 26-2. Data validation***). One clock pulse is generated for each data bit transferred.

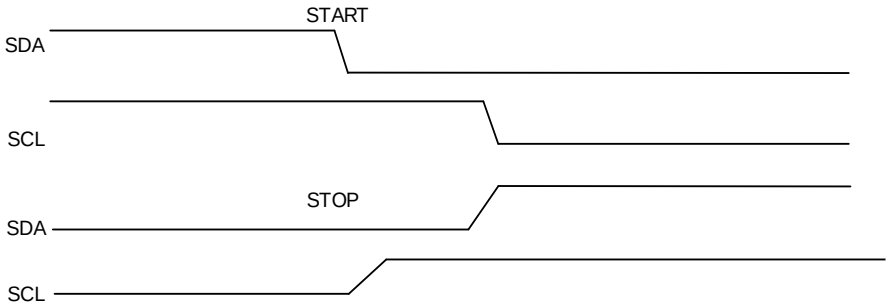
**Figure 26-2. Data validation**




#### START and STOP signal

All transactions begin with a START and are terminated by a STOP (see ***Figure 26-3. START and STOP condition***). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START signal. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP signal.

**Figure 26-3. START and STOP condition**




Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device. It operates in slave mode by default. When it generates a START signal, the interface automatically switches from slave to master. If an arbitration loss or a STOP generation occurs, then the interface switches from master to slave, allowing multimaster capability.

An I2C slave will continue to detect addresses after a START signalon I2C bus and compare the detected address with its slave address which is programmable by software. Once the two addresses match, the I2C slave will send an ACK to the I2C bus and responses to the following command on I2C bus: transmitting or receiving the desired data. Additionally, if General Call is enabled by software, the I2C slave always responses to a General Call Address (0x00). The I2C block support both 7-bit and 10-bit address modes.

Data and addresses are transferred as 8-bit bytes, MSB first. The first byte(s) following the START signalcontain the address (one in 7-bit mode, two in 10-bit mode). The address is always transmitted in master mode.

A 9th clock pulse follows the 8 clock cycles of byte transmission, during which the receiver must send an acknowledge bit to the transmitter. Acknowledge can be enabled or disabled by software.

An I2C master always initiates or end a transfer using START or STOP signaland it's also responsible for SCL clock generation.

In master mode, if AUTOEND=1, the STOP signalis generated automatically by hardware. If AUTOEND=0, the STOP signalgenerated by software, or the master can generate a RESTART signalto start a new transfer.

**Figure 26-4. I2C communication flow with 10-bit address (Master Transmit)**

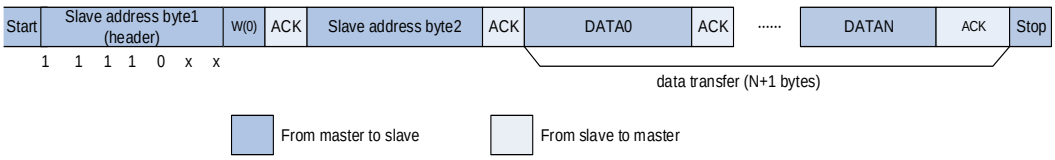


**Figure 26-5. I2C communication flow with 7-bit address (Master Transmit)**

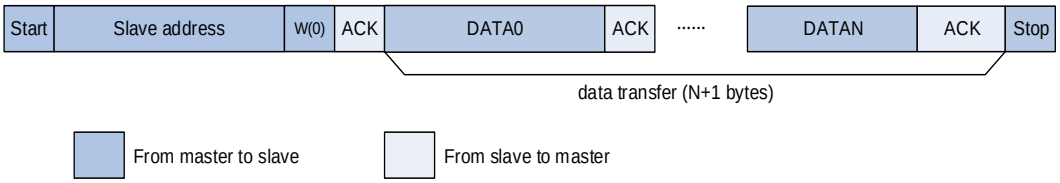
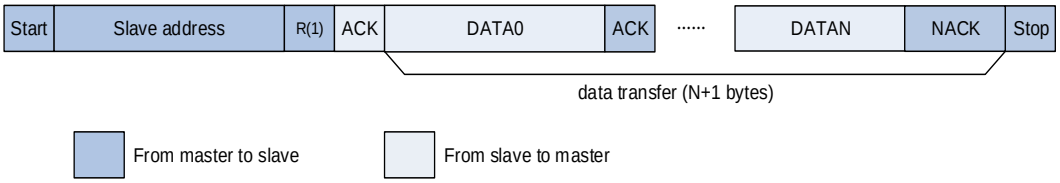



**Figure 26-6. I2C communication flow with 7-bit address (Master Receive)**




In 10-bit addressing mode, the HEAD10R bit can configured to decide whether the complete address sequence must be executed, or only the header to be sent. When HEAD10R=0, the complete 10 bit address read sequence must be excuted with START + header of 10-bit address in write direction + slave address byte 2 + RESTART + header of 10-bit address in read direction, as is shown in ***Figure 26-7. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0)***.

In 10-bit addressing mode, if the master reception follows a master transmission between the same master and slave, the address read sequence can be RESTART + header of 10-bit address in read direction, as is shown in ***Figure 26-8. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)***.

**Figure 26-7. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0)**

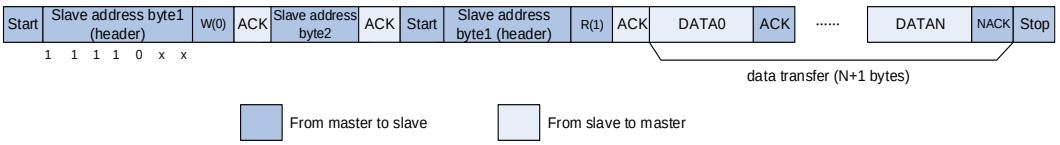
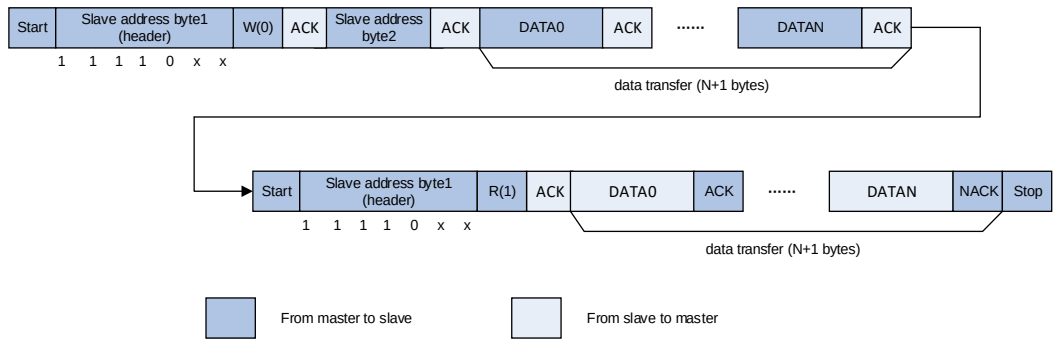



**Figure 26-8. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)**



### 26.3.3. Noise filter

The noise filters must be configured before setting the I2CEN bit in I2C_CTL0 register if it is necessary. The analog noise filter is disabled by setting the ANOFF bit in I2C_CTL0 register and enabled when ANOFF is 0. It can suppress spikes with a pulse width up to 130ns in fast mode and fast mode plus.

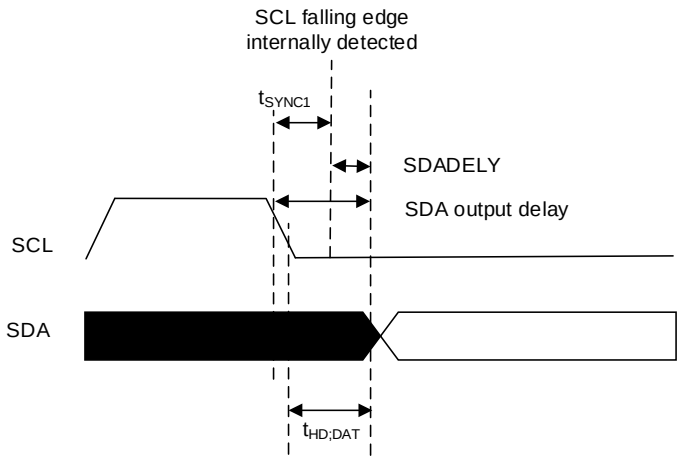
The digital noise filter can be used by configuring the DNF[3:0] bit in I2C_CTL0 register. The the level of the SCL or the SDA will not be changed if the level is stable for no more than DNF[3:0]×$t_{I2CCLK}$. The length of spikes to be suppressed is configured by DNF[3:0].

### 26.3.4. I2C timings configuration

The PSC[3:0], SCLDELY[3:0] and SDADELY[3:0] bits in the I2C_TIMING register must be configured in order to guarantee a correct data hold and setup time used in I2C communication.

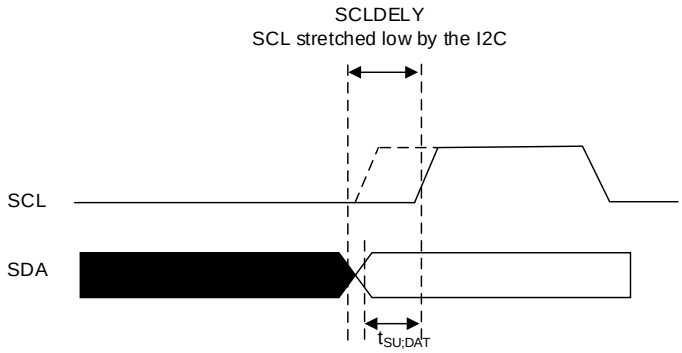
If the data is already available in I2C_TDATA register, the data will be sent on SDA after the SDADELY delay. As is shown in ***Figure 26-9. Data hold time***.

**Figure 26-9. Data hold time**




The SCLDELY counter starts when the data is sent on SDA output. As is shown in ***Figure 26-10. Data setup time***.

**Figure 26-10. Data setup time**




When the SCL falling edge is internally detected, a delay is inserted before sending SDA output. This delay is $t_{SDADELY}$ = SDADELY * $t_{PSC}$ + $t_{I2CCLK}$ where $t_{PSC}$ = (PSC+1) * $t_{I2CCLK}$. $t_{SDADELY}$ effects $t_{HD;DAT}$. The total delay of SDA output is $t_{SYNC1}$ + {[SDADELY * (PSC+1) + 1] * $t_{I2CCLK}$}. $t_{SYNC1}$ depends on SCL falling slope, the delay of analog filter, the delay of digital filter and delay of SCL synchronization to I2CCLK clock. The delay of SCL synchronization to I2CCLK clock is 2 to 3 $t_{I2CCLK}$.

**Note:** When SDADELY = 0, $t_{SDADELY}$ = $t_{PSC}$.

SDADELY must match condition as follows:

- SDADELY≥{$t_f$(max)+$t_{HD;DAT}$(min)-$t_{AF}$(min)-[(DNF+3)*$t_{I2CCLK}$]}/[(PSC+1)*$t_{I2CCLK}$]
- SDADELY≤{$t_{HD;DAT}$(max)-$t_{AF}$(max)-[(DNF+4)*$t_{I2CCLK}$]}/[(PSC+1)*$t_{I2CCLK}$]

**Note:** $t_{AF}$ is the delay of analog filter. The $t_{HD;DAT}$ should be less than the maximum of $t_{VD;DAT}$.

When SS = 0, after $t_{SDADELY}$ delay, the slave had to stretch the clock before the data writing to I2C_TDATA register, SCL is low during the data setup time. The setup time is $t_{SCLDELY}$=(SCLDELY+1)*$t_{PSC}$. $t_{SCLDELY}$ effects $t_{SU;DAT}$.

SCLDELY must match condition as follows:

- SCLDELY≥{[$t_r$(max)+$t_{SU;DAT}$(min)]/[(PSC+1)*$t_{I2CCLK}$]}-1

In master mode, the SCL clock high and low levels must be configured by programming the PSC[3:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_TIMING register.

When the SCL falling edge is internally detected, a delay is inserted before releasing the SCL output. This delay is $t_{SCLL}$=(SCLL+1)*$t_{PSC}$ where $t_{PSC}$=(PSC+1)*$t_{I2CCLK}$. $t_{SCLL}$ impacts the SCL low time $t_{LOW}$.

When the SCL rising edge is internally detected, a delay is inserted before forcing the SCL output to low level. This delay is $t_{SCLH}$=(SCLH+1)*$t_{PSC}$ where $t_{PSC}$=(PSC+1)*$t_{I2CCLK}$. $t_{SCLH}$ impacts the SCL high time $t_{HIGH}$.

**Note:** When the I2C is enabled, the timing configuration and SS mode must not be changed.

**Table 26-2. Data setup time and data hold time**

| Symbol | Parameter | Standard mode | | Fast mode | | Fast mode plus | | SMBus | | Unit |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | Min | Max | Min | Max | |
| $t_{HD;DAT}$ | Data hold time | 0 | - | 0 | - | 0 | - | 0.3 | - | us |
| $t_{VD;DAT}$ | Data valid time | - | 3.45 | - | 0.9 | - | 0.45 | - | - | |
| $t_{SU;DAT}$ | Data setup time | 250 | - | 100 | - | 50 | - | 250 | - | ns |
| $t_r$ | Rising time of SCL and SDA | - | 1000 | - | 300 | - | 120 | - | 1000 | |
| $t_f$ | falling time of SCL and SDA | - | 300 | - | 300 | - | 120 | - | 300 | |

### 26.3.5. I2C reset

A software reset can be performed by clearing the I2CEN bit in the I2C_CTL0 register. When a software reset is generated, the SCL and SDA are released. The communication control bits and status bits come back to the reset value. Software reset have no effect on configuration registers. The impacted register bits are START, STOP, NACKEN in I2C_CTL1 register, I2CBSY, TBE, TI, RBNE, ADDSEND, NACK, TCR, TC, STPDET, BERR, LOSTARB and OUERR in I2C_STAT register. Additionally, when the SMBus is supported, PECTRANS in I2C_CTL1 register, PECERR, TIMEOUT and SMBALT in I2C_STAT are also impacted.

In order to perform the software reset, I2CEN must be kept low during at least 3 APB clock cycles. This is ensured by writing software sequence as follows:

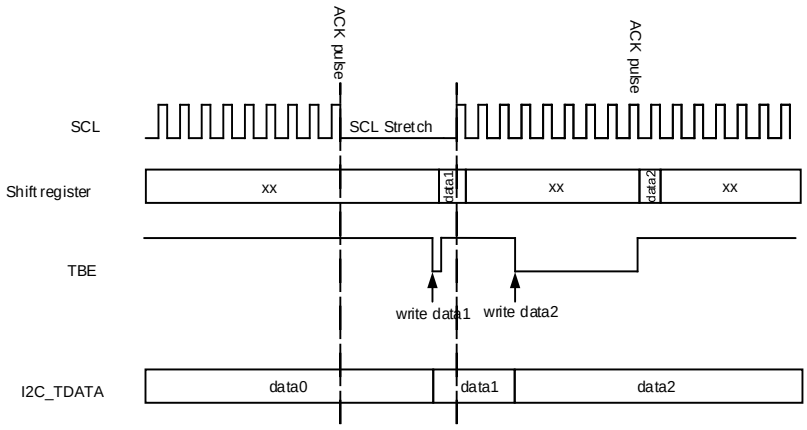
- Write I2CEN = 0
- Check I2CEN = 0
- Write I2CEN = 1

### 26.3.6. Data transfer

#### Data Transmission

When transmitting data,, if TBE is 0, it indicates that the I2C_TDATA register is not empty, the data in I2C_TDATA register is moved to the shift register after the 9th SCL pulse. Then the data will be transmitted through the SDA line from the shift register. If TBE is 1, it indicates that the I2C_TDATA register is empty, the SCL line is stretched low until I2C_TDATA is not empty. The stretch begins after the 9th SCL pulse.

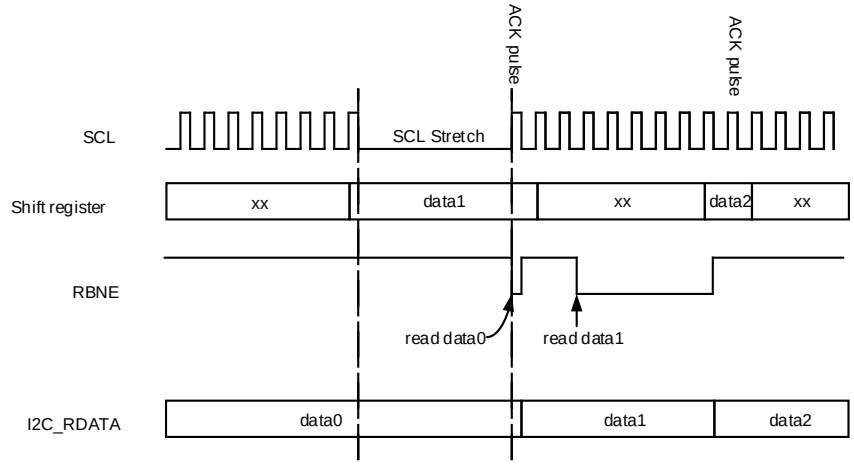
**Figure 26-11. Data transmission**




## Data Reception

When receiving data, the data will be received in the shift register first. If RBNE is 0, the data in the shift register will move into I2C_RDATA register. If RBNE is 1, the SCL line wii be stretched until the previous received data in I2C_RDATA is read. The stretch is inserted before the acknowledge pulse.

**Figure 26-12. Data reception**




## Reload and automatic end mode

In order to manage byte transfer and to shut down the communication in modes as is shown in ***Table 26-3. Communication modes to be shut down***, the I2C embedded a byte counter in the hardware.

**Table 26-3. Communication modes to be shut down**

| Working mode | Action |
|---|---|
| Master mode | NACK, STOP and RESTART generation |
| Slave receiver mode | ACK control |

| Working mode | Action |
|---|---|
| SMBus mode | PEC generation/checking |

The number of bytes to be transferred is configured by BYTENUM[7:0] in I2C_CTL1 register. If BYTENUMis greater than 255, or in slave byte control mode, the reload mode must be enabled by setting the RELOAD bit in I2C_CTL1 register. In reload mode, when BYTENUM counts to 0, the TCR bit will be set, and an interrupt will be generated if TCIE is set. Once the TCR flag is set, SCL is stretched. The TCR bit is cleared by writing a non-zero number in BYTENUM.

**Note:** The reload mode must be disabled after the last reloading of BYTENUM[7:0].

The reload mode must be disabled when the automatic end mode is enabled. In automatic end mode, the master will send a STOP signal automatically when the BYTENUM[7:0] counts to 0.

### 26.3.7. I2C slave mode

#### Initialization

When works in slave mode, at least one slave address should be enabled. Slave address 1 can be programmed in I2C_SADDR0 register and slave address 2 can be programmed in I2C_SADDR1 register. ADDRESSEN in I2C_SADDR0 register and ADDRESS2EN in I2C_SADDR1 register should be set when the corresponding address is used. 7-bit address or 10-bit address can be programmed in ADDRESS[9:0] in I2C_SADDR0 register by configuring the ADDFORMAT bit in 7-bit address or 10-bit address.

The ADDM[6:0] in I2C_CTL2 register defines which bits of ADDRESS[7:1] are compared with an incoming address byte, and which bits are ignored.

The ADDMSK2[2:0] is used to mask ADDRESS2[7:1] in I2C_SADDR1 register. For details, refer to the description of ADDMSK2[2:0] in I2C_SADDR1 register.

When the I2C received address matches one of its enabled addresses, the ADDSEND will be set, and an interrupt is generated if the ADDMIE bit is set. The READDR[6:0] bits in I2C_STAT register will store the received address. And TR bit in I2C_STAT register updates after the ADDSEND is set. The bit will let the slave to know whether to act as a transmitter or receiver.

#### SCL line stretching

The clock stretching is used in slave mode by default (SS=0), the SCL line can be stretched low if necessary. The SCL will be stretched in following cases.

- The SCL is stretched when the ADDSEND bit is set, and released when the ADDSEND bit is cleared.
- In slave transmitting mode, after the ADDSEND bit is cleared, the SCL will be stretched before the first data byte writing to the I2C_TDATA register. Or the SCL will be stretched before the new data is written to the I2C_TDATA register after the previous data

transmission is completed.

- In slave receiving mode, a new reception is completed but the data in I2C_RDATA register has not been read.
- When SBCTL=1 and RELOAD=1, after the transfer of the last byte, TCR is set. Before the TCR is cleared, the SCL will be stretched.
- The I2C stretches SCL low during $[(SDADELY+SCLDELY+1)*(PSC+1)+1]*t_{I2CCLK}$ after detecting the SCL falling edge.

The clock stretching can be disabled by setting the SS bit in I2C_CTL0 register (SS=1). The SCL will not be stretched in following cases.

- When the ADDSEND is set, the SCL will be not stretched.
- In slave transmitting mode, before the first SCL pulse, the data should be written in the I2C_TDATA register . Or else the OUERR bit in the I2C_STAT register will be set, if the ERRIE bit is set, an interrupt will be generated. When the STPDET bit is set and the first data transmission starts, OUERR bit in the I2C_STAT register will also be set.
- In slave receiving mode, before the 9th SCL pulse (ACK pulse) occurred by the next data byte, the data must be read out from the I2C_RDATA register. Or else the OUERR bit in the I2C_STAT register will be set, if the ERRIE bit is set, and an interrupt will be generated.

### Slave byte control mode

In slave receiving mode, the slave byte control mode can be enabled by setting the SBCTL bit in the I2C_CTL0 register to allow byte ACK control. When SS=1, the slave byte control mode is not allowed.

When using slave byte control mode, the reload mode must be enabled by setting the RELOAD bit in I2C_CTL1 register. In slave byte control mode, BYTENUM[7:0] in I2C_CTL1 register must be configured as 1 in the ADDSEND interrupt service routine and reloaded to 1 after each byte received. The TCR bit in I2C_STAT register will be set when a byte is received, the SCL will be stretched low by slave between the 8th and 9th clock pulses. Then the data can be read from the I2C_RDATA register, and the slave determines to send an ACK or a NACK by configuring the NACKEN bit in the I2C_CTL1 register. When the BYTENUM[7:0] is written a non-zero value, the slave will release the stretch.

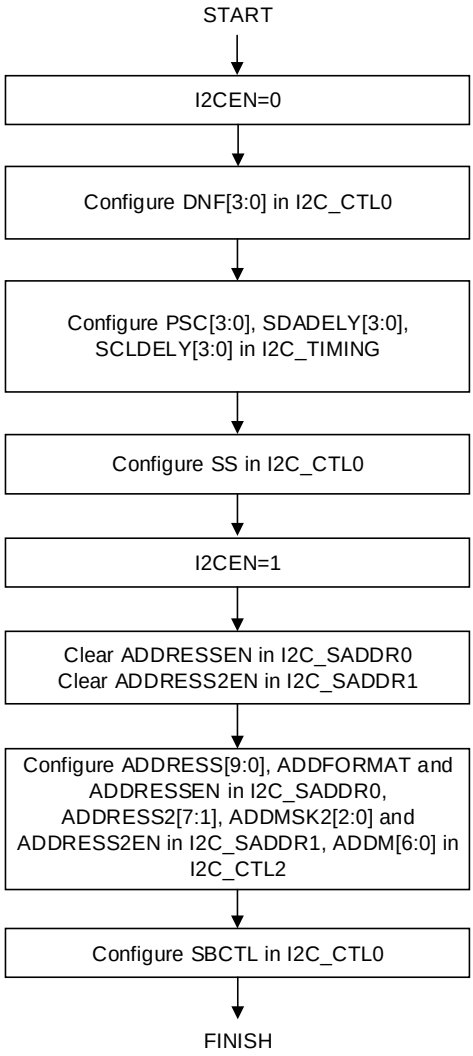
When the BYTENUM[7:0] is greater than 0x1, there is no stretch between the reception of two data bytes.

**Note:** The SBCTL bit can be configured in following cases:

1. I2CEN=0.
2. The slave has not been addressed.
3. ADDSEND=1.

Only when the ADDSEND=1 or TCR=1, the RELOAD bit can be modified.

**Figure 26-13. I2C initialization in slave mode**




### Programming model in slave transmitting mode

When the I2C_TDATA register is empty, the TI bit in I2C_STAT register will be set. If the TIE bit in I2C_CTL0 register is set, an interrupt will be generated. The NACK bit in I2C_STAT register will be set when a NACK is received. And an interrupt is generated if the NACKIE bit is set in the I2C_CTL0 register. The TI bit in I2C_STAT register will not be set when a NACK is received.

The STPDET bit in I2C_STAT register will be set when a STOP is received. If the STPDETIE in I2C_CTL0 register is set, an interrupt will be generated.

When SBCTL is 0, if ADDSEND=1, and the TBE bit in I2C_STAT register is 0, the data in I2C_TDATA register can be chosed to be transmitted or flushed. The data is flushed by setting the TBE bit.

When SBCTL=1, the slave works in slave byte control mode, the BYTENUM[7:0] must be configured in the ADDSEND interrupt service routine. And the number of TI events is equal to the value of BYTENUM[7:0].

When SS=1, the SCL will not be stretched when ADDSEND bit in I2C_STAT register is set. In this case, the data in I2C_TDATA register can not be flushed in ADDSEND interrupt service routine. So the first byte to be sent must be programmed in the I2C_TDATA register previously.

- This data can be the one which is written in the last TI event of the last transfer.
- Setting the TBE bit can flush the data if it is not the one to be sent, , then a new byte can be written in I2C_TDATA register. The STPDET must be 0 when the data transmission begins. Or else the OUERR bit in I2C_STAT register will be set.
- When interrupt or DMA is used in slave transmitter, if a TI event is needed, in order to generate a TI event both the TI bit and the TBE bit must be set.

**Figure 26-14. Programming model for slave transmitting when SS=0**

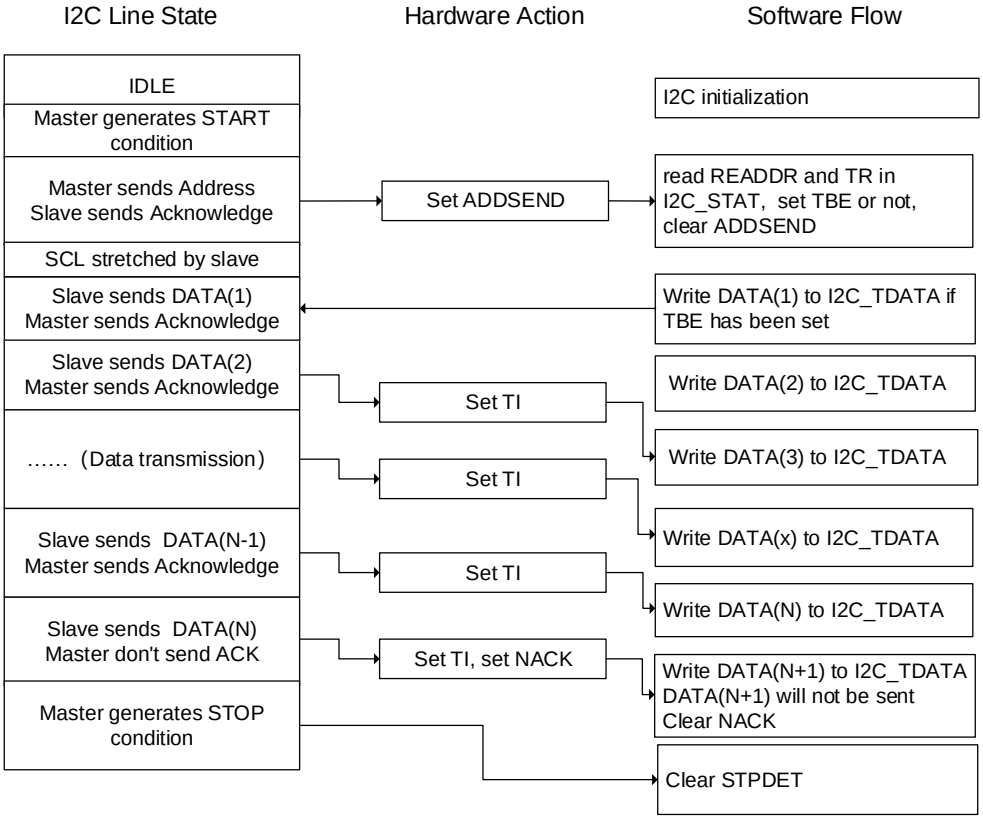
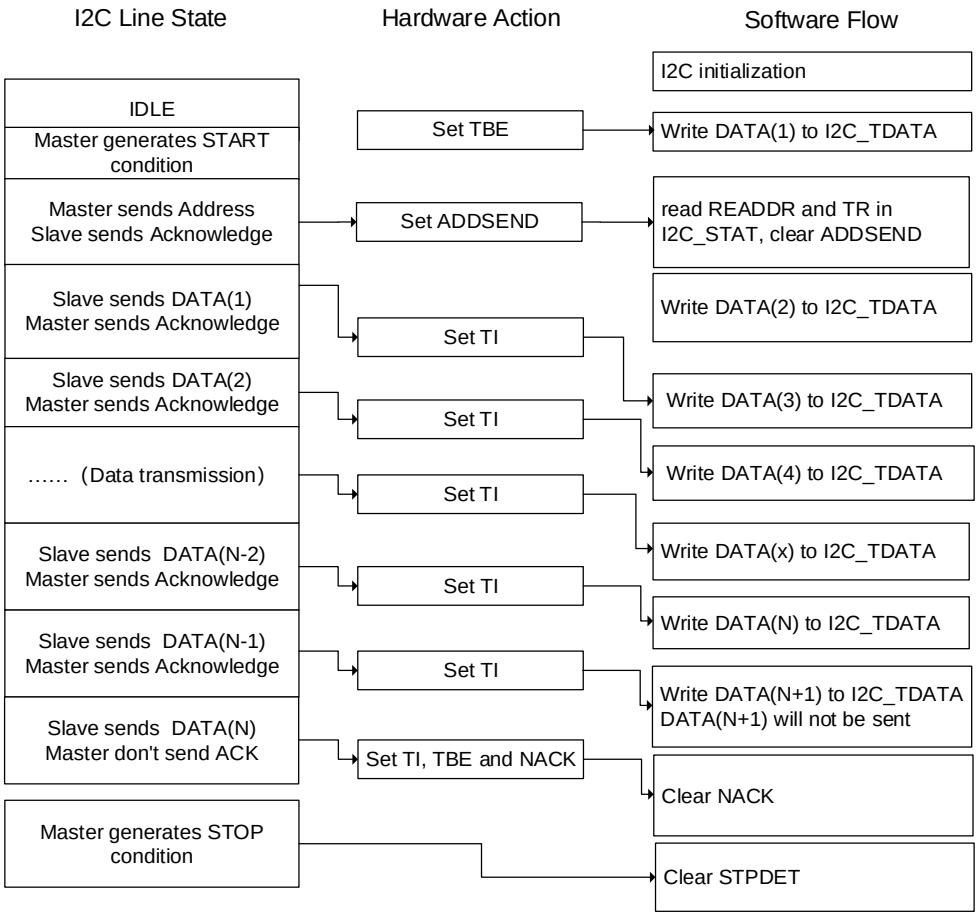


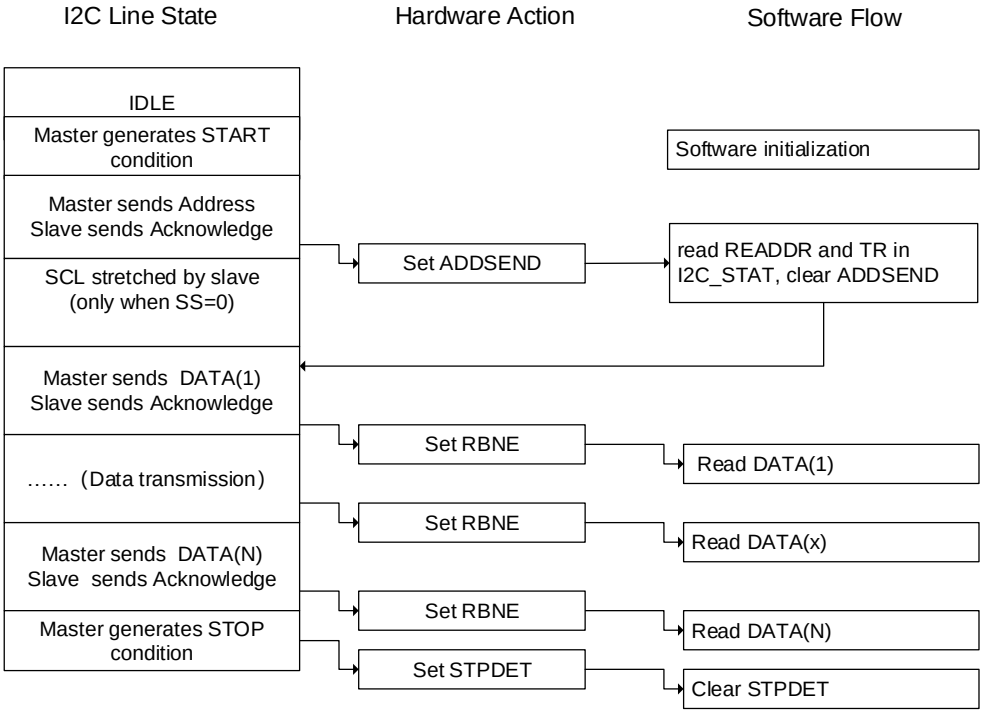
**Figure 26-15. Programming model for slave transmitting when SS=1**




### Programming model in slave receiving mode

When the I2C_RDATA is not empty, the RBNE bit in I2C_STAT register is set, and if the RBNEIE bit in I2C_CTL0 register is set, an interrupt will be generated. When a STOP is received, STPDET will be set in I2C_STAT register. If the STPDETIE bit in I2C_CTL0 register is set, and an interrupt will be generated.

**Figure 26-16. Programming model for slave receiving**




## 26.3.8. I2C master mode

### Initialization

The SCLH[7:0] and SCLL[7:0] in I2C_TIMING register should be configured when I2CEN is 0. In order to support multi-master communication and slave clock stretching, a clock synchronization mechanism is implemented.

The SCLL[7:0] and SCLH[7:0] are used for the low level counting and high level couting respectively. After a $t_{SYNC1}$ delay, when the SCL low level is detected, the SCLL[7:0] starts counting, if the SCLL[7:0] in I2C_TIMING register is reached by SCLL[7:0] counter, the I2C will release the SCL clock. After a $t_{SYNC2}$ delay, when the SCL high level is detected, the SCLH[7:0] starts counting, if the SCLH[7:0] in I2C_TIMING register is reached by SCLH[7:0] counter, the I2C will stretch the SCL clock.

So the master clock period is:

$t_{SCL}=t_{SYNC1}+t_{SYNC2}+\{[(SCLH[7:0]+1)+(SLLL[7:0]+1)]*(PSC+1)*t_{I2CCLK}\}$.

The $t_{SYNC1}$ depends on the SCL falling slope, delay by input analog and digital noise filter and SCL synchronization with I2CCLK clock, which generally 2 to 3 I2CCLK periods. The $t_{SYNC2}$ depends on the SCL rising slope, delay by input analog and digital noise filter and SCL synchronization with I2CCLK clock, which generally 2 to 3 I2CCLK periods. The delay by digital noise filter is DNF[3:0]*$t_{I2CCLK}$.

When works in master mode, the ADD10EN bit, SADDRESS[9:0] bits, TRDIR bit should be configured in I2C_CTL1 register. When the addressing mode is 10-bit in master receiving

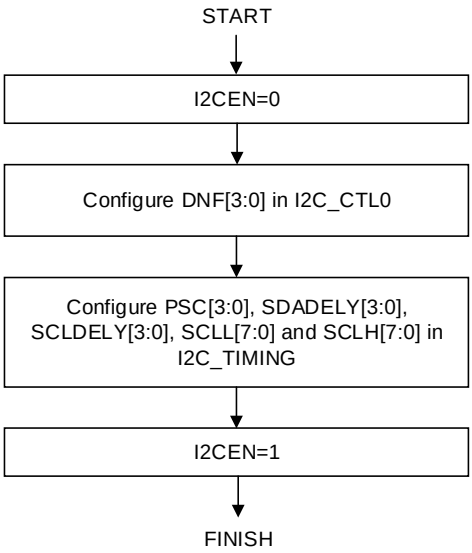
mode, the HEAD10R bit must be configured to decide whether the complete address sequence must be executed, or only the header to be sent. The number of bytes to be transferred should be configured in BYTENUM[7:0] in I2C_CTL1 register. If the number of bytes to be transferred is equal to or greater than 255, BYTENUM[7:0] should be configured as 0xFF. Then the master sends the START signal. All the bits above should be configured before the START is set. The slave address will be sent after the START signal when the I2CBSY bit I2C_STAT register is detected as 0. When the arbitration is lost, the master changes to slave mode and the START bit will be cleared by hardware. When the slave address has been sent, the START bit will be cleared by hardware.

**Note**：BYTENUM[7:0] can not be configured as 0.

In 10-bit addressing mode, if the master receives a NACK after the transmission of 10-bit header, the master will resend it until ACK is received. The ADDSENDC bit must be set to stop sending the slave address.

If the START bit is set, meanwhile the ADDSEND is set by addressing as a slave, the master changes to slave mode. The ADDSENDC bit must be set to clear the START bit.

**Figure 26-17. I2C initialization in master mode**




### Programming model in master transmitting mode

In master transmitting mode, the TI bit is set after the ACK is received of each byte transmission. If the TIE bit in I2C_CTL0 register is set, an interrupt will be generated. The bytes to be transferred is programmed in BYTENUM[7:0] in I2C_CTL1 register. If the bytes to be transferred is greater than 255, RELOAD bit in I2C_CTL1 register must be set to enable the reload mode. In reload mode, when data of BYTENUM[7:0] bytes have been transferred, the TCR bit in I2C_STAT register will be set and the SCL stretches unitil BYTENUM[7:0] is modified with a non-zero value.

When a NACK is received, the TI bit will not set.

- If data of BYTENUM[7:0] bytes have been transferred and RELOAD=0, the AUTOEND bit in I2C_CTL1 can be set to generate a STOP signal automatically. When AUTOEND is 0, the TC bit in I2C_STAT register will be set and the SCL is stretched. In this case, the master can generate a STOP signal by setting the STOP bit in the I2C_CTL1 register. Or generate a RESTART signalto start a new transfer. The TC bit is cleared when the START / STOP bit is set.
- If a NACK is received, a STOP signal is automatically generated, the NACK is set in I2C_STAT register, if the NACKIE bit is set, an interrupt will be generated.

**Note:** When the RELOAD bit is 1, the AUTOEND has no effect.

**Figure 26-18. Programming model for master transmitting (N<=255)**

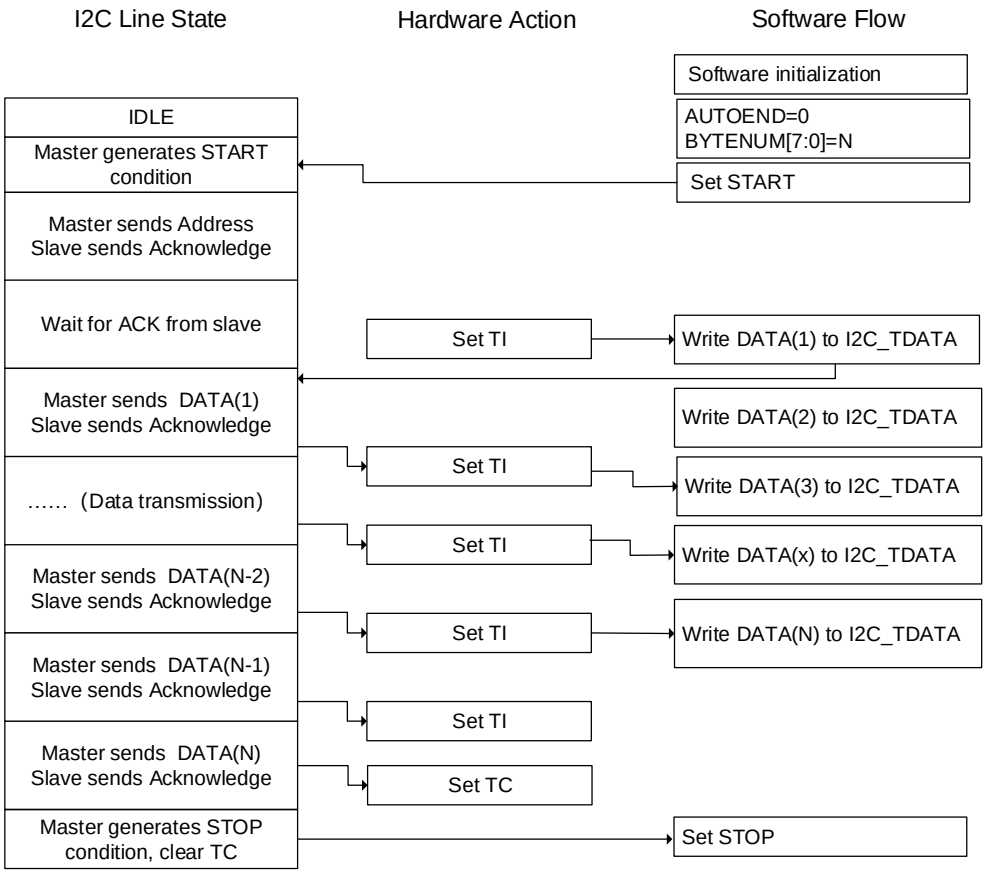
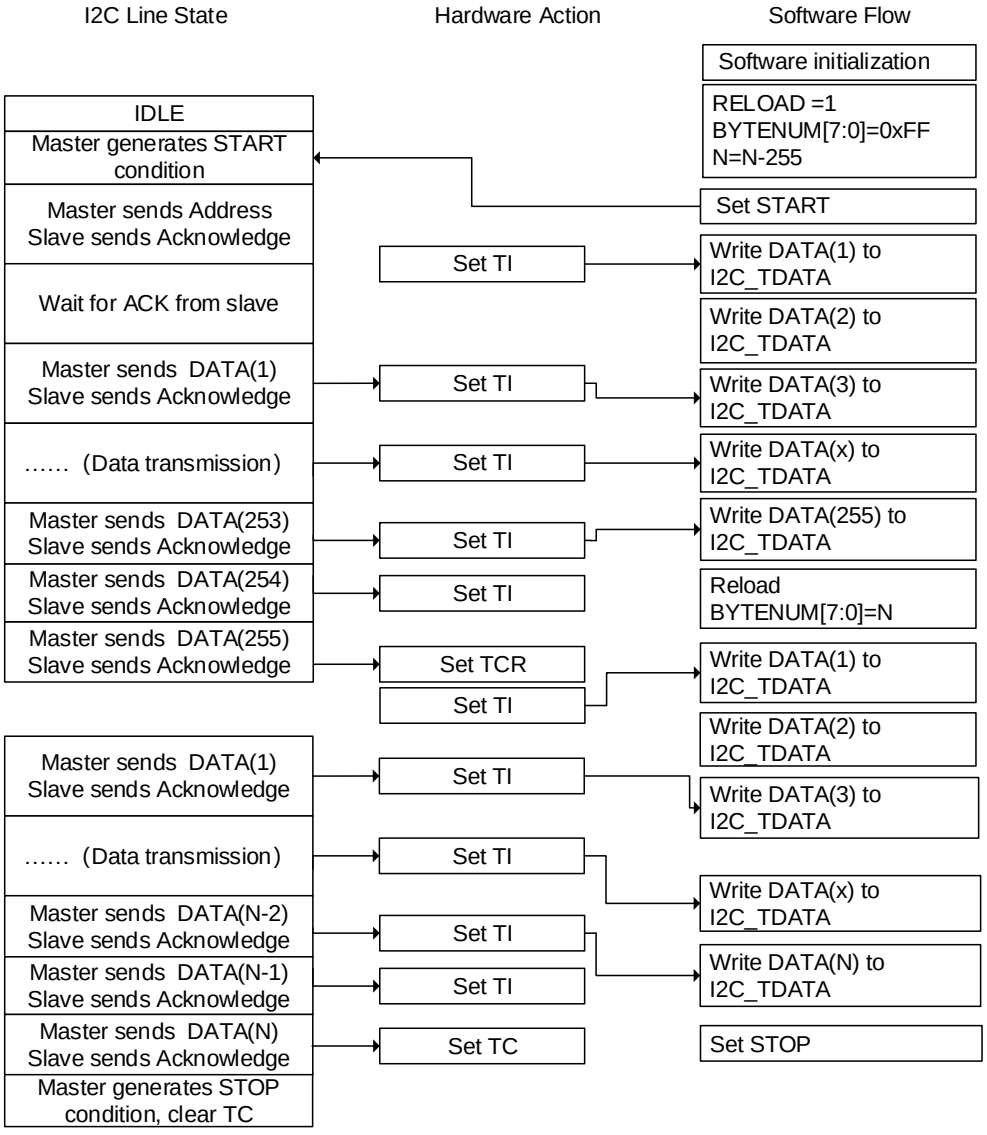


**Figure 26-19. Programming model for master transmitting (N>255)**




## Programming model in master receiving mode

In master receiving mode, the RBNE bit in I2C_STAT register will be set when a byte is received. If the RBNEIE bit is set in I2C_CTL0 register, an interrupt will be generated. If the number of bytes to be received is greater than 255, RELOAD bit in I2C_CTL1 register must be set to enable the reload mode. In reload mode, when data of BYTENUM[7:0] bytes have been transferred, the TCR bit in I2C_STAT register will be set and the SCL stretches unitil BYTENUM[7:0] is modified with a non-zero value.

If data of BYTENUM[7:0] bytes have been transferred and RELOAD=0, the AUTOEND bit in I2C_CTL1 can be set to generate a STOP signal automatically. When AUTOEND is 0, the TC bit in I2C_STAT register will be set and the SCL is stretched. In this case, the master can generate a STOP signalby setting the STOP bit in the I2C_CTL1 register. Or generate a RESTART signal to start a new transfer. The TC bit is cleared when the START / STOP bit is set.

**Figure 26-20. Programming model for master receiving (N<=255)**

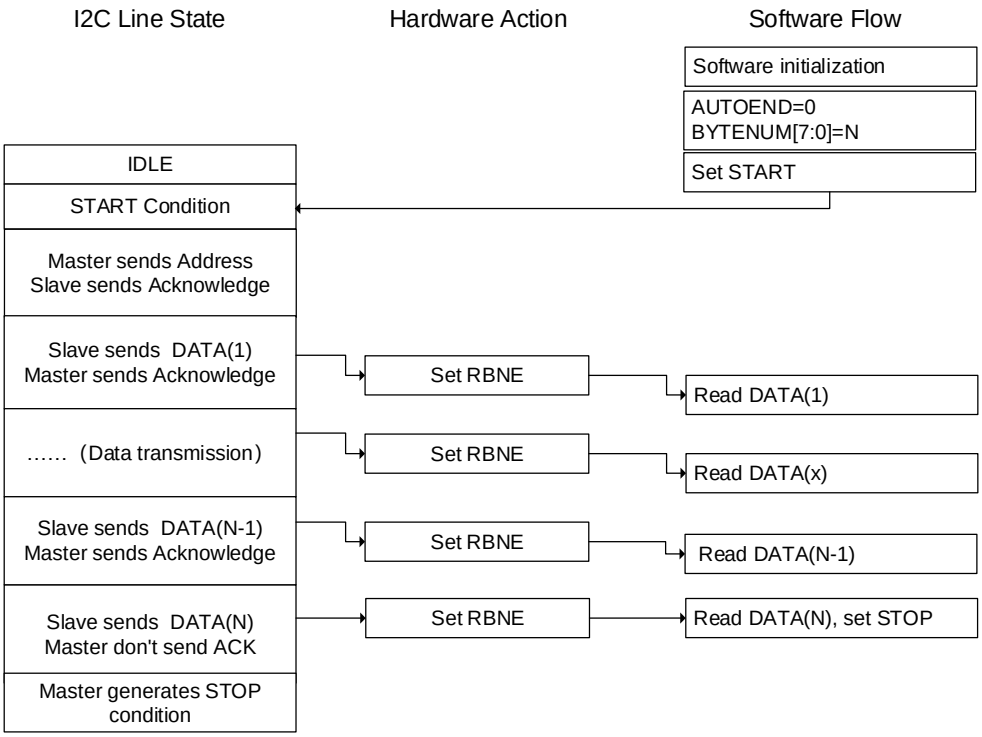
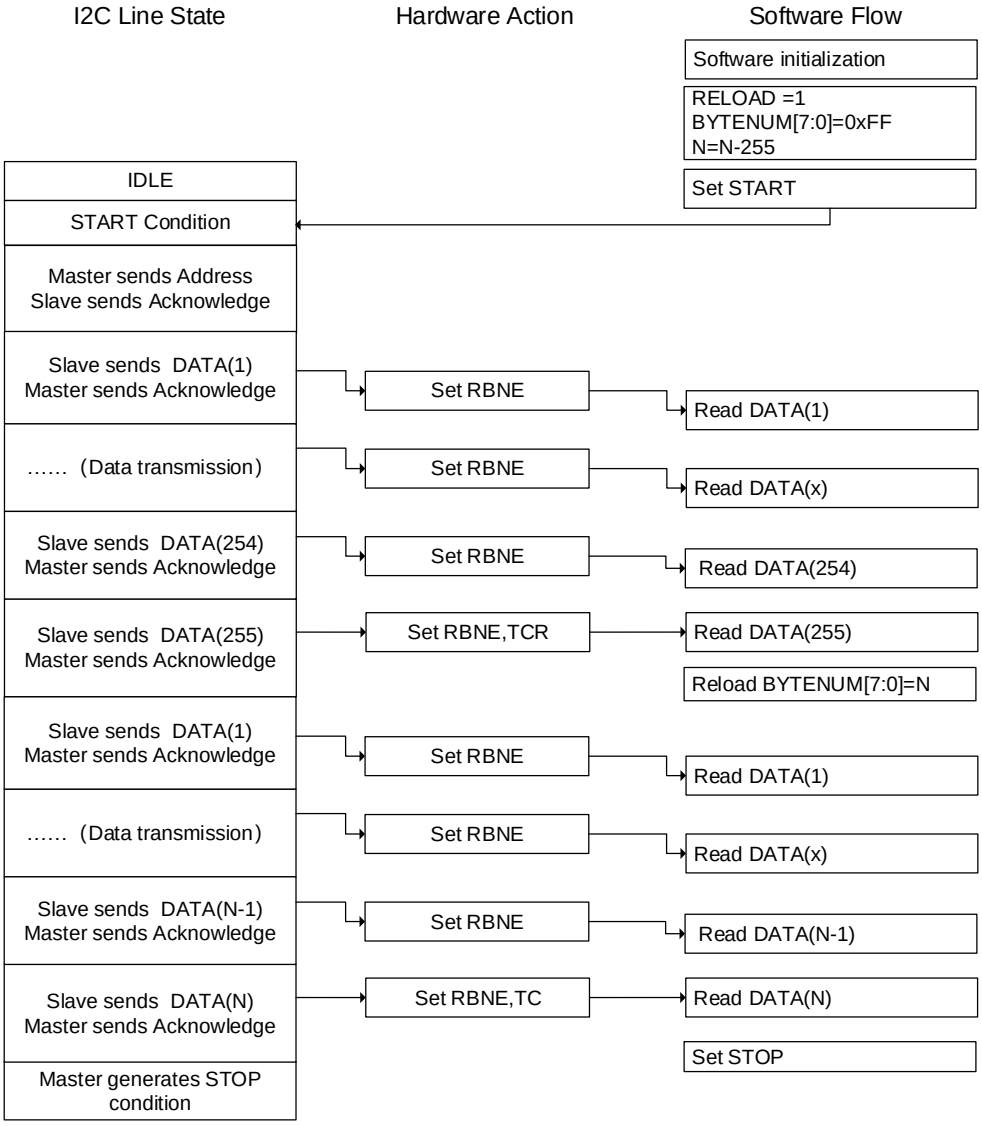


**Figure 26-21. Programming model for master receiving (N>255)**




## 26.3.9. SMBus support

The System Management Bus (abbreviated to SMBus or SMB) is a single-ended simple two-wire bus for the purpose of lightweight communication. Most commonly it is found in computer motherboards for communication with power source for ON/OFF instructions. It is derived from I2C for communication with low-bandwidth devices on a motherboard, especially power related chips such as a laptop's rechargeable battery subsystem (see Smart Battery Data).

### SMBus protocol

Each message transaction on SMBus follows the format of one of the defined SMBus protocols. The SMBus protocols are a subset of the data transfer formats defined in the I2C specifications. I2C devices that can be accessed through one of the SMBus protocols are compatible with the SMBus specifications. I2C devices that do not adhere to these protocols cannot be accessed by standard methods as defined in the SMBus and Advanced Configuration and Power Management Interface (abbreviated to ACPI) specifications.

### Address resolution protocol

The SMBus uses I2C hardware and I2C hardware addressing, but adds second-level software for building special systems. Additionally, its specifications include an Address Resolution Protocol that can make dynamic address allocations. Dynamic reconfiguration of the hardware and software allow bus devices to be 'hot-plugged' and used immediately, without restarting the system. The devices are recognized automatically and assigned unique addresses. This advantage results in a plug-and-play user interface. In this protocol there is a very useful distinction between a system host and all the other devices in the system, that is the host provides address assignment function.

### SMBus slave byte control

The slave byte control of SMBus receiver is the same as I2C. It allows the ACK control of each byte. Slave byte control mode is enabled by setting SBCTL bit in I2C_CTL0 register.

### Host notify protocol

When the SMBHAEN bit in the I2C_CTL0 register is set, the SMBus supports the host notify protocol. In this protocol, the device acts as a master and the host as a slave, and the host will acknowledge the SMBus host address.

### Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency of 10 kHz to prevent locking up the bus. I2C can be a 'DC' bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This will notify to the master that the slave is busy but does not want to lose the communication. The slave device will allow continuation after its task is completed. There is no limit in the I2C bus protocol as to how long this delay can be, whereas for a SMBus system, it would be limited to 25~35ms. SMBus protocol just assumes that if something takes too long, then it means that there is a problem on the bus and that all devices must reset in order to clear this mode. Slave devices are not allowed to hold the clock low too long.

The timeout detection can be enabled by setting TOEN and EXTOEN bits in the I2C_TIMEOUT register. The timer must be configured to guarantee that the timeout detected before the maximum time given in the SMBus specification.

The value programmed in BUSTOA[11:0] is used to check the $t_{TIMEOUT}$ parameter. To detect the SCL low level timeout, the TOIDLE bit must be 0. And the timer can be enabled by setting the TOEN bit in the I2C_TIMEOUT register, after the TOEN bit is set, the BUSTOA[11:0] and the TOIDLE bit cannot be changed. If the low level time of SCL is greater than (BUSTOA+1)*2048*$t_{I2CCLK}$, the TIMEOUT flag will be set in I2C_STAT register.

The BUSTOB[11:0] is used to check the $t_{LOW:SEXT}$ of the slave and the $t_{LOW:MEXT}$ of the master. The timer can be enabled by setting the EXTOEN bit in the I2C_TIMEOUT register,

after the EXTOEN bit is set, the BUSTOB[11:0] cannot be changed. If the SCL stretching time of the SMBus peripheral is greater than (BUSTOB+1)*2048*$t_{I2CCLK}$ and within the timeout range described in the bus idle detection section, the TIMEOUT bit in the I2C_STAT register will be set.

### Packet error checking

There is a CRC-8 calculator in I2C block to perform Packet Error Checking for I2C data. A PEC (packet error code) byte is appended at the end of each transfer. The byte is calculated as CRC-8 checksum, calculated over the entire message including the address and read/write bit. The polynomial used is $x^8+x^2+x+1$ (the CRC-8-ATM HEC algorithm, initialized to zero).

When I2C is disabled, the PEC can be enabled by setting the PECEN bit in I2C_CTL0 register.Since the PEC transmission is managed by BYTENUM[7:0] in I2C_CTL1 register, SBCTL bit must be set when act as a slave. When PECTRANS is set and the RELOAD bit is cleared, PEC is transmitted after the BYTENUM[7:0]-1 data byte. The PECTRANS has no effect if RELOAD is set.

### SMBus alert

The SMBus has an extra optional shared interrupt signal called SMBALERT# which can be used by slaves to tell the host to ask its slaves about events of interest. The host processes the interrupt and accesses all SMBALERT# devices through the Alert Response Address at the same time. If the SMBALERT# is pulled low by the devices, the devices will acknowledge the Alert Response Address. When SMBHAEN is 0, it is configured as a slave device, the SMBA pin will be pulled low by setting the SMBALTEN bit in the I2C_CTL0 register. Meanwhile the Alert Response Address is enabled. When SMBHAEN is 1, it is configured as a host, and the SMBALTEN is 1, as soon as a falling edge is detected on the SMBA pin, the SMBALT flag will be set in the I2C_STAT register. If the ERRIE bit is set in the I2C_CTL0 register, an interrupt will be generated. When SMBALTEN is 0, the level of ALERT line is considered high even if the SMBA pin is low. The SMBA pin can be used as a standard GPIO if SMBALTEN is 0.

### Bus idle detection

If the master detects that the high level duration of the clock and data signals is greater than $t_{HIGH,MAX}$, the bus can be considered idle.

This timing parameter includes the case of a master that has been dynamically added to the bus and may not have detected a state transition on a SMBCLK or SMBDAT lines. In this case, in order to ensure that there is no ongoing transmission, the master must wait long enough.

The BUSTOA[11:0] bits must be programmed with the timer reload value to enable the $t_{IDLE}$ check in order to obtain the $t_{IDLE}$ parameter. To detect SCL and SDA high level timeouts, the TOIDLE bit must be set. Then setting the TOEN bit in the I2C_TIMEOUT register to enable the timer, after the TOEN bit is set, the BUSTOA[11:0] bit and the TOIDLE bit cannot be

changed. If the high level time of both SCL and SDA is greater than (BUSTOA+1)*4*$t_{I2CCLK}$, the TIMEOUT flag will be set in the I2C_STAT register.

#### SMBus slave mode

The SMBus receiver must be able to NACK each command or data it receives. For ACK control in slave mode, slave byte control mode can be enabled by setting SBCTL bit in I2C_CTL0 register.

SMBus-specific addresses should be enabled when needed. The SMBus Device Default address (0b1100 001) is enabled by setting the SMBDAEN bit in the I2C_CTL0 register. The SMBus Host address (0b0001 000) is enabled by setting the SMBHAEN bit in the I2C_CTL0 register. The Alert Response Address (0b0001 100) is enabled by setting the SMBALTEN bit in the I2C_CTL0 register.

### 26.3.10. SMBus mode

#### SMBus master transmitter and slave receiver

The PEC in SMBus master mode can be transmitted by setting the PECTRANS bit before setting the START bit, and the number of bytes in the BYTENUM[7:0] field must be configured. In this case, the total number of transmissions when TI interrupt occur is BYTENUM-1. So if BYTENUM=0x1 and PECTRANS bit is set, the data in I2C_PEC register will be transmitted automatically. If AUTOEND is 1 the SMBus master will send the STOP signal after the PEC byte automatically. If the AUTOEND is 0, the SMBus master can send a RESTART signal after the PEC. The PEC byte in I2C_PEC register will be sent after BYTENUM-1 bytes, and the TC flag will be set after PEC is sent, then the SCL line is stretched low. The RESTART must be set in the TC interrupt routine.

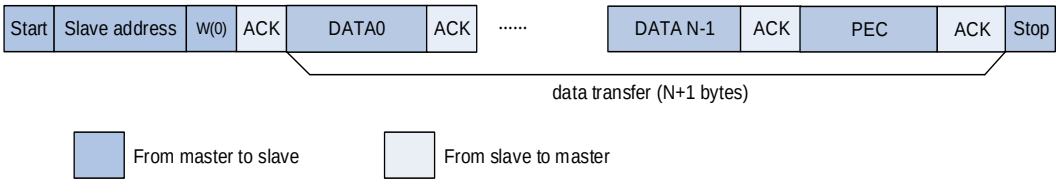
When used as slave receiver, in order to allow PEC checking at the end of the number of bytes transmitted, SBCTL must be set. To configure ack control for each byte, the RELOAD must be set to enable the RELOAD mode. In order to check the PEC byte, it is necessary to clear the RELOAD bit and set PECTRANS bit. After receiving BYTENUM-1 data, the next received byte will be compared with the data in the I2C_PEC register. If the PEC values does not match, the NACK is automatically generated. If the PEC values matches, the ACK is automatically generated, regardless of the NACKEN bit value. When PEC byte is received, it is also copied into the I2C_RDATA register, and RBNE flag will be set. If the ERRIE bit in I2C_CTL0 register is 1, when PEC value does not match, the PECERR flag will be set and the interrupt will be generated. If ACK control is not required, then PECTRANS can be set to 1 and BYTENUM can be programmed according to the number of bytes to be received.

**Note:**

1. After the RELOAD bit is set, the PECTRANS cannot be changed.

2. When SMBus acts as a slave transmitter and uses the PEC function, BYTENUM[0:7] must be configured before detecting the START signal.

**Figure 26-22. SMBus master transmitter and slave receiver communication flow**




### SMBus master receiver and slave transmitter

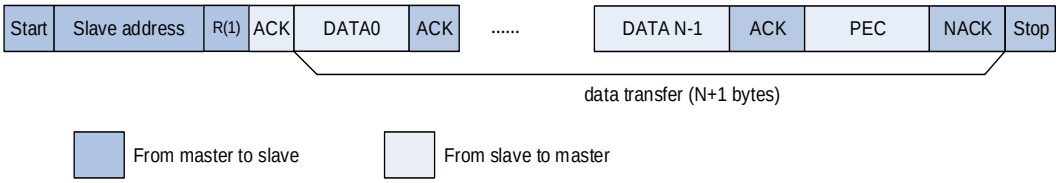
If the SMBus master is required to receive PEC at the end of bytes transfer, automatic end mode can be enabled. Before sending a START condition on the bus, PECTRANS bit must be set and slave addresses must be programmed. After receiving BYTENUM-1 data, the next received byte will be compared with the data in the I2C_PEC register automatically. A NACK is respond to the PEC byte before STOP condition.

If the SMBus master receiver is required to generate a RESTART signal after receiving PEC byte, automatic end mode must be disabled. Before sending a START signal to the bus, PECTRANS bit must be set and slave addresses must be programmed. After receiving BYTENUM-1 data, the next received byte will be compared with the data in the I2C_PEC register automatically. The TC flag will be set after PEC is sent, then the SCL line is stretched low. The RESTART can be set in the TC interrupt routine.

When used as slave transmitter, in order to allow PEC transmission at the end of BYTENUM[7:0] bytes, SBCTL must be set. If PECTRANS bit is set, the number of bytes in BYTENUM[7:0] contains PEC byte. In this case, if the number of bytes requested by the master is greater than BYTENUM-1, the total number of TI interrupts will be BYTENUM-1, and the data in the I2C_PEC register will be transmitted automatically.

**Note:** After the RELOAD bit is set, the PECTRANS cannot be changed.

**Figure 26-23. SMBus master receiver and slave transmitter communication flow**




## 26.3.11. Wakeup from power saving mode

When the address of I2C matches correctly, it can wake up from MCU sleep mode and Deep-sleep mode (APB clock is off). In order to wake up from power saving mode, WUEN bit must be set in the I2C_CTL0 register and the IRC64M must be selected as the clock source for I2CCLK. During power saving mode, the IRC64M is switched off. The I2C interface switches the IRC64M on, and stretches SCL low until IRC64M is woken up when a START is detected. Then the IRC64M is used as the clock of I2C to receive the address. When address matching is detected, I2C stretches SCL during MCU wake-up. The SCL is released until the software clears the ADDSEND flag and the transmission proceeds normally. If the detected address

does not match, IRC64M will be closed again and the MCU will not be wake up.

Only an address match interrupt (ADDMIE = 1) can wakeup the MCU. If the clock source of I2C is the system clock, or WUEN = 0, IRC64M will not switched on after receiving start signal. When wakeup from power saving mode is enabled, the digital filter must be disabled and the SS bit in I2C_CTL0 must be cleared. Before entering power saving mode (I2CEN = 0), the I2C peripheral must be disabled if wakeup from power saving mode is disabled (WUEN = 0).

### 26.3.12. Use DMA for data transfer

As is shown in I2C slave mode and I2C master mode, each time TI or RBNE is asserted, software should write or read a byte, this may cause CPU's high overload. The DMA controller can be used to process TI and RBNE flag: each time TI or RBNE is asserted, DMA controller does a read or write operation automatically.

The DMA transmission request is enabled by setting the DENT bit in I2C_CTL0 register. The DMA reception request is enabled by setting the DENR bit in I2C_CTL0 register. In master mode, the slave address, transmission direction, number of bytes and START bit are programmed by software. The DMA must be initialized before setting the START bit. The number of bytes to be transferred is configured in the BYTENUM[7:0] in I2C_CTL1 register. In slave mode, the DMA must be initialized before the address match event or in the ADDSEND interrupt routine, before clearing the ADDSEND flag.

### 26.3.13. I2C error and interrupts

The I2C error flags are listed in ***Table 26-4. I2C error flags***.

**Table 26-4. I2C error flags**

| I2C Error Name | Description |
|---|---|
| BERR | Bus error |
| LOSTARB | Arbitration lost |
| OUERR | Overrun / Underrun flag |
| PECERR | CRC value doesn't match |
| TIMEOUT | Bus timeout in SMBus mode |
| SMBALT | SMBus Alert |

The I2C interrupt events and flags are listed in ***Table 26-5. I2C interrupt events***.

**Table 26-5. I2C interrupt events**

| Interrupt event | Event flag | Enable control bit |
|---|---|---|
| I2C_RDATA is not empty during receiving | RBNE | RBNEIE |
| Transmit interrupt | TI | TIE |
| STOP siganl detected in slave mode | STPDET | STPDETIE |
| Transfer complete reload | TCR | TCIE |
| Transfer complete | TC | |
| Address match | ADDSEND | ADDMIE |

| Interrupt event | Event flag | Enable control bit |
|---|---|---|
| Not acknowledge received | NACK | NACKIE |
| Bus error | BERR | ERRIE |
| Arbitration Lost | LOSTARB | |
| Overrun/Underrun error | OUERR | |
| PEC error | PECERR | |
| Timeout error | TIMEOUT | |
| SMBus Alert | SMBALT | |

### 26.3.14. I2C debug mode

When the microcontroller enters the debug mode (Cortex®-M7 core halted), the SMBus timeout either continues to work normally or stops, depending on the I2Cx_HOLD configuration bits in the DBG module.

## 26.4. Register definition

I2C0 base address: 0x4000 5400

I2C1 base address: 0x4000 5800

I2C2 base address: 0x4000 C000

I2C3 base address: 0x4000 5C00

### 26.4.1. Control register 0 (I2C_CTL0)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | PECEN | SMBALTEN | SMBDAEN | SMBHAEN | GCEN | WUEN | SS | SBCTL |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DENR | DENT | Reserved | ANOFF | DNF[3:0] | | | | ERRIE | TCIE | STPDETIE | NACKIE | ADDMIE | RBNEIE | TIE | I2CEN |
| rw | rw | | rw | rw | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | PECEN | PEC Calculation Switch<br>0: PEC Calculation off<br>1: PEC Calculation on |
| 22 | SMBALTEN | SMBus Alert enable<br>0: SMBA pin is not pulled down (device mode) or SMBus Alert pin SMBA is disabled (host mode)<br>1: SMBA pin is pulled down (device mode) or SMBus Alert pin SMBA is enabled (host mode) |
| 21 | SMBDAEN | SMBus device default address enable<br>0: Device default address is disabled, the default address 0b1100001x will be not acknowledged.<br>1: Device default address is enabled, the default address 0b1100001x will be acknowledged. |
| 20 | SMBHAEN | SMBus Host address enable<br>0: Host address is disabled, address 0b0001000x will be not acknowledged.<br>1: Host address is enabled, address 0b0001000x will be acknowledged. |

| | | |
|---|---|---|
| 19 | GCEN | Whether or not to response to a General Call (0x00)<br>0: Slave won't response to a General Call<br>1: Slave will response to a General Call |
| 18 | WUEN | Wakeup from power saving mode enable<br>This bit is cleared when mcu wakeup from power saving mode.<br>0: Wakeup from power saving mode disable.<br>1: Wakeup from power saving mode enable.<br>**Note:** WUEN can be set only when DNF[3:0] = 0000. |
| 17 | SS | Whether to stretch SCL low when data is not ready in slave mode.<br>This bit is set and cleared by software.<br>0: SCL Stretching is enabled<br>1: SCL Stretching is disabled<br>**Note:** When in master mode, this bit must be 0. This bit can be modified when I2CEN = 0. |
| 16 | SBCTL | Slave byte control<br>This bit is used to enable hardware byte control in slave mode.<br>0: Slave byte control is disabled<br>1: Slave byte control is enabled |
| 15 | DENR | DMA enable for reception<br>0: DMA is disabled for reception<br>1: DMA is enabled for reception |
| 14 | DENT | DMA enable for transmission<br>0: DMA is disabled for transmission<br>1: DMA is enabled for transmission |
| 13 | Reserved | Must be kept at reset value. |
| 12 | ANOFF | Analog noise filter disable<br>0: Analog noise filter is enabled<br>1: Analog noise filter is disabled<br>**Note:** This bit can only be programmed when the I2C is disabled (I2CEN = 0). |
| 11:8 | DNF[3:0] | Digital noise filter<br>0000: Digital filter is disabled<br>0001: Digital filter is enabled and filter spikes with a length of up to 1 $t_{I2CCLK}$<br>...<br>1111: Digital filter is enabled and filter spikes with a length of up to15 $t_{I2CCLK}$<br>These bits can only be modified when the I2C is disabled (I2CEN = 0). |
| 7 | ERRIE | Error interrupt enable<br>0: Error interrupt disabled<br>1: Error interrupt enabled. When BERR, LOSTARB, OUERR, PECERR, TIMEOUT or SMBALT bit is set, an interrupt will be generated. |

| | | |
|---|---|---|
| 6 | TCIE | Transfer complete interrupt enable<br>0: Transfer complete interrupt is disabled<br>1: Transfer complete interrupt is enabled |
| 5 | STPDETIE | Stop detection interrupt enable<br>0: Stop detection (STPDET) interrupt is disabled<br>1: Stop detection (STPDET) interrupt is enabled |
| 4 | NACKIE | Not acknowledge received interrupt enable<br>0: Not acknowledge (NACK) received interrupt is disabled<br>1: Not acknowledge (NACK) received interrupt is enabled |
| 3 | ADDMIE | Address match interrupt enable in slave mode<br>0: Address match (ADDSEND) interrupt is disabled<br>1: Address match (ADDSEND) interrupt is enabled |
| 2 | RBNEIE | Receive interrupt enable<br>0: Receive (RBNE) interrupt is disabled<br>1: Receive (RBNE) interrupt is enabled |
| 1 | TIE | Transmit interrupt enable<br>0: Transmit (TI) interrupt is disabled<br>1: Transmit (TI) interrupt is enabled |
| 0 | I2CEN | I2C peripheral enable<br>0: I2C is disabled<br>1: I2C is enabled |

### 26.4.2. Control register 1 (I2C_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | PECTRANS | AUTOEND | RELOAD | BYTENUM[7:0] | | | | | | | |
| | | | | | rw | rw | rw | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NACKEN | STOP | START | HEAD10R | ADD10EN | TRDIR | SADDRESS[9:0] | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | PECTRANS | PEC Transfer<br>Set by software. |

| | | |
|---|---|---|
| | | Cleared by hardware in the following cases:<br>When PEC byte is transferred or ADDSEND bit is set or STOP signal is detected or I2CEN=0.<br>0: Don't transfer PEC value<br>1: Transfer PEC<br>**Note:** This bit has no effect when RELOAD=1, or SBCTL=0 in slave mode. |
| 25 | AUTOEND | Automatic end mode in master mode<br>0: TC bit is set when the transfer of BYTENUM[7:0] bytes is completed.<br>1: a STOP signal is sent automatically when the transfer of BYTENUM[7:0] bytes is completed.<br>**Note:** This bit works only when RELOAD=0. This bit is set and cleared by software. |
| 24 | RELOAD | Reload mode<br>0: After the data of BYTENUM[7:0] bytes transfer, the transfer is completed.<br>1: After data of BYTENUM[7:0] bytes transfer, the transfer is not completed and the new BYTENUM[7:0] will be reloaded. Every time when the BYTENUM[7:0] bytes have been transferred, the TCR bit in I2C_STAT register will be set.<br>This bit is set and cleared by software. |
| 23:16 | BYTENUM[7:0] | Number of bytes to be transferred<br>These bits are programmed with the number of bytes to be transferred. When SBCTL=0, these bits have no effect.<br>**Note:** When the START bit is set to 1, there bits cannot be modified, and the software cannot change this bit during transmission. |
| 15 | NACKEN | Generate NACK in slave mode<br>0: an ACK is sent after receiving a new byte.<br>1: a NACK is sent after receiving a new byte.<br>**Note:** The bit can be set by software, and cleared by hardware when the NACK is sent, or when a STOP signal is detected or ADDSEND is set, or when I2CEN=0. When PEC is enabled, whether to send an ACK or a NACK is not depend on the NACKEN bit. When SS=1, and the OUERR bit is set, the value of NACKEN is ignored and a NACK will be sent. |
| 14 | STOP | Generate a STOP signal on I2C bus<br>This bit is set by software and cleared by hardware when I2CEN=0 or STOP condition is detected.<br>0: STOP will not be sent<br>1: STOP will be sent |
| 13 | START | Generate a START condition on I2C bus<br>This bit is set by software and cleared by hardware after the address is sent. When the arbitration is lost, or a timeout error occurred, or I2CEN=0, this bit can also be cleared by hardware. It can be cleared by software by setting the ADDSENDC bit in I2C_STATC register.<br>0: START will not be sent |

| | | |
|---|---|---|
| | | 1: START will be sent |
| 12 | HEAD10R | 10-bit address header executes read direction only in master receive mode<br>0: The 10 bit master receive address sequence is START + header of 10-bit address (write) + slave address byte 2 + RESTART + header of 10-bit address (read).<br>1: The 10 bit master receive address sequence is RESTART + header of 10-bit address (read).<br>**Note:** When the START bit is set to 1, this bit cannot be modified, and the software cannot change this bit during transmission. |
| 11 | ADD10EN | 10-bit addressing mode enable in master mode<br>0: 7-bit addressing in master mode<br>1: 10-bit addressing in master mode<br>**Note:** When the START bit is set to 1, this bit cannot be modified, and the software cannot change this bit during transmission. |
| 10 | TRDIR | Transfer direction in master mode<br>0: Master transmit<br>1: Master receive<br>**Note:** When the START bit is set to 1, this bit cannot be modified, and the software cannot change this bit during transmission. |
| 9:0 | SADDRESS[9:0] | Slave address to be sent<br>SADDRESS[9:8]: Slave address bit 9:8<br>If ADD10EN = 0, these bits have no effect.<br>If ADD10EN = 1, these bits should be written with bits 9:8 of the slave address to be sent.<br>SADDRESS[7:1]: Slave address bit 7:1<br>If ADD10EN = 0, these bits should be written with the 7-bit slave address to be sent.<br>If ADD10EN = 1, these bits should be written with bits 7:1 of the slave address to be sent.<br>SADDRESS0: Slave address bit 0<br>If ADD10EN = 0, this bit has no effect.<br>If ADD10EN = 1, this bit should be written with bit 0 of the slave address to be sent<br>**Note:** When the START bit is set to 1, this bit cannot be modified, and the software cannot change this bit during transmission. |

### 26.4.3. Slave address register 0 (I2C_SADDR0)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| ADDRESSEN | Reserved | ADDFORMAT | ADDRESS[9:8] | ADDRESS[7:1] | ADDRESS0 |
|---|---|---|---|---|---|
| rw | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | ADDRESSEN | I2C address enable<br>0: I2C address disable.<br>1: I2C address enable. |
| 14:11 | Reserved | Must be kept at reset value. |
| 10 | ADDFORMAT | Address mode for the I2C slave<br>0: 7-bit address<br>1: 10-bit address<br>**Note:** When ADDRESSEN is set, this bit should not be written. |
| 9:8 | ADDRESS[9:8] | Highest two bits of a 10-bit address<br>**Note:** When ADDRESSEN is set, this bit should not be written. |
| 7:1 | ADDRESS[7:1] | 7-bit address or bits 7:1 of a 10-bit address<br>**Note:**<br>1. When ADDRESSEN is set, this bit should not be written.<br>2. When the slave address is set to the reserved address 0b'1111000, the slave will not respond if an address match occurs. |
| 0 | ADDRESS0 | Bit 0 of a 10-bit address<br>**Note:** When ADDRESSEN is set, this bit should not be written. |

## 26.4.4. Slave address register 1 (I2C_SADDR1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDRESS2EN | Reserved | | | | ADDMSK2[2:0] | | | ADDRESS2[7:1] | | | | | | | Reserved |
| rw | | | | | rw | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | ADDRESS2EN | Second I2C address enable |

| | | |
|---|---|---|
| | | 0: Second I2C address disable.<br>1: Second I2C address enable. |
| 14:11 | Reserved | Must be kept at reset value. |
| 10:8 | ADDMSK2[2:0] | ADDRESS2[7:1] mask<br>Defines which bits of ADDRESS2[7:1] are compared with an incoming address byte, and which bits are masked (don't care).<br>000: No mask, all the bits must be compared.<br>N(001~110): ADDRESS2[n:1] is masked. Only ADDRESS2[7:n+1] are compared.<br>111: ADDRESS2[7:1] are masked. All 7-bit received addresses are acknowledged except the reserved address (0b0000xxx and 0b1111xxx).<br>**Note:** When ADDRESS2EN is set, these bits should not be written. If ADDMSK2 is not equal to 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged even if all the bits are matched. |
| 7:1 | ADDRESS2[7:1] | Second I2C address for the slave<br>**Note:**<br>1. When ADDRESS2EN is set, these bits should not be written.<br>2. When the slave address is set to the reserved address 0b'1111000, the slave will not respond if an address match occurs. |
| 0 | Reserved | Must be kept at reset value. |

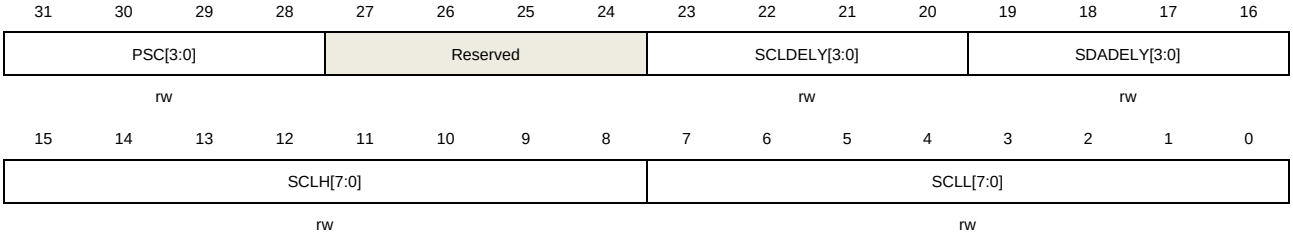
## 26.4.5. Timing register (I2C_TIMING)

Address offset: 0x10

Reset value: 0x0000 0000

This register cannot be modified when the I2C is enable (I2CEN=1).

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | PSC[3:0] | Timing prescaler<br>In order to generate the clock period $t_{PSC}$ used for data setup and data hold counters, these bits are used to configure the prescaler for I2CCLK. The $t_{PSC}$ is also used for SCL high and low level counters.<br>$t_{PSC}=(PSC+1)*t_{I2CCLK}$ |
| 27:24 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 23:20 | SCLDELY[3:0] | Data setup time<br>A delay $t_{SCLDELY}$ between SDA edge and SCL rising edge can be generated by configuring these bits. And during $t_{SCLDELY}$, the SCL line is stretched low in master mode and in slave mode when SS = 0.<br>$t_{SCLDELY}=(SCLDELY+1)*t_{PSC}$ |
| 19:16 | SDADELY[3:0] | Data hold time<br>A delay $t_{SDADELY}$ between SCL falling edge and SDA edge can be generated by configuring these bits. And during $t_{SDADELY}$, the SCL line is stretched low in master mode and in slave mode when SS = 0.<br>0x0: $t_{SDADELY}$ = 1*$t_{PSC}$<br>0x1-0xF: $t_{SDADELY}$=SDADELY*$t_{PSC}$ |
| 15:8 | SCLH[7:0] | SCL high period<br>SCL high period can be generated by configuring these bits.<br>$t_{SCLH}=(SCLH+1)*t_{PSC}$<br>**Note:** These bits can only be used in master mode. |
| 7:0 | SCLL[7:0] | SCL low period<br>SCL low period can be generated by configuring these bits.<br>$t_{SCLL}=(SCLL+1)*t_{PSC}$<br>**Note:** These bits can only be used in master mode. |

## 26.4.6. Timeout register (I2C_TIMEOUT)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXTOEN | Reserved | | | BUSTOB[11:0] | | | | | | | | | | | |
| rw | | | | rw | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TOEN | Reserved | | TOIDLE | BUSTOA[11:0] | | | | | | | | | | | |
| rw | | | rw | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | EXTOEN | Extended clock timeout detection enable<br>When a cumulative SCL stretch time is greater than $t_{LOW:EXT}$, a timeout error will be occurred. $t_{LOW:EXT}$ = (BUSTOB +1) x 2048 x $t_{I2CCLK}$.<br>0: Extended clock timeout detection is disabled.<br>1: Extended clock timeout detection is enabled. |
| 30:28 | Reserved | Must be kept at reset value. |
| 27:16 | BUSTOB[11:0] | Bus timeout B<br>Configure the cumulative clock extension timeout. |

| | | |
|---|---|---|
| | | In master mode, the master cumulative clock low extend time $t_{LOW:MEXT}$ is detected.<br>In slave mode, the slave cumulative clock low extend time $t_{LOW:SEXT}$ is detected.<br>$t_{LOW:EXT}$=(BUSTOB+1)*2048*$t_{I2CCLK}$.<br>**Note:** These bits can be modified only when EXTOEN =0. |
| 15 | TOEN | Clock timeout detection enable<br>If the SCL stretch time greater than $t_{TIMEOUT}$ when TOIDLE =0 or high for more than $t_{IDLE}$ when TOIDLE =1, a timeout error is detected.<br>0: SCL timeout detection is disabled<br>1: SCL timeout detection is enabled |
| 14:13 | Reserved | Must be kept at reset value. |
| 12 | TOIDLE | Idle clock timeout detection<br>0: BUSTOA is used to detect SCL low timeout<br>1: BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle<br>**Note:** This bit can be written only when TOEN =0. |
| 11:0 | BUSTOA[11:0] | Bus timeout A<br>When TOIDLE = 0, $t_{TIMEOUT}$=(BUSTOA+1)*2048*$t_{I2CCLK}$.<br>When TOIDLE = 1, $t_{IDLE}$=(BUSTOA+1)*4*$t_{I2CCLK}$.<br>**Note:** These bits can be written only when TOEN =0. |

### 26.4.7. Status register (I2C_STAT)

Address offset: 0x18

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | READDR[6:0] | | | | | | | TR |
| | | | | | | | | r | | | | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I2CBSY | Reserved | SMBALT | TIMEOUT | PECERR | OUERR | LOSTARB | BERR | TCR | TC | STPDET | NACK | ADDSEND | RBNE | TI | TBE |
| r | | r | r | r | r | r | r | r | r | r | r | r | r | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:17 | READDR[6:0] | Received match address in slave mode<br>When the ADDSEND bit is set, these bits store the matched address. In the case of a 10-bit address, READDR[6:0] stores the header of the 10-bit address followed by the 2 MSBs of the address. |
| 16 | TR | Whether the I2C is a transmitter or a receiver in slave mode |

| | | |
|---|---|---|
| | | This bit is updated when the ADDSEND bit is set.<br>0: Receiver<br>1: Transmitter |
| 15 | I2CBSY | Busy flag<br>This bit is set by hardware when a START signal is detected and cleared by hardware after a STOP signal. When I2CEN=0, this bit is also cleared by hardware.<br>0: No I2C communication.<br>1: I2C communication active. |
| 14 | Reserved | Must be kept at reset value. |
| 13 | SMBALT | SMBus Alert<br>When SMBHAEN=1, SMBALTEN=1, and a SMBALERT event (falling edge) is detected on SMBA pin, this bit will be set by hardware. It is cleared by software by setting the SMBALTC bit. This bit is cleared by hardware when I2CEN=0.<br>0: SMBALERT event is not detected on SMBA pin<br>1: SMBALERT event is detected on SMBA pin |
| 12 | TIMEOUT | TIMEOUT flag.<br>When a timeout or extended clock timeout occurred, this bit will be set. It is cleared by software by setting the TIMEOUTC bit and cleared by hardware when I2CEN=0.<br>0: no timeout or extended clock timeout occur<br>1: a timeout or extended clock timeout occur |
| 11 | PECERR | PEC error<br>This flag is set by hardware when the received PEC does not match with the content of I2C_PEC register. Then a NACK is automatically sent. It is cleared by software by setting the PECERRC bit and cleared by hardware when I2CEN=0.<br>0: Received PEC and content of I2C_PEC match<br>1: Received PEC and content of I2C_PEC don’t match, I2C will send NACK regardless of NACKEN bit. |
| 10 | OUERR | Overrun/Underrun error in slave mode<br>In slave mode with SS=1, when an overrun/underrun error occurs, this bit will be set by hardware. It is cleared by software by setting the OUERRC bit and cleared by hardware when I2CEN=0.<br>0: No overrun or underrun occurs<br>1: Overrun or underrun occurs |
| 9 | LOSTARB | Arbitration Lost<br>It is cleared by software by setting the LOSTARBC bit and cleared by hardware when I2CEN=0.<br>0: No arbitration lost.<br>1: Arbitration lost occurs and the I2C block changes back to slave mode. |
| 8 | BERR | Bus error<br>When an unexpected START or STOP signal on I2C bus is detected, a bus error |

| | | |
|---|---|---|
| | | occurs and this bit will be set. It is cleared by software by setting BERRC bit and cleared by hardware when I2CEN=0.<br>0: No bus error<br>1: A bus error detected |
| 7 | TCR | Transfer complete reload<br>This bit is set by hardware when RELOAD=1 and data of BYTENUM[7:0] bytes have been transferred. It is cleared by software when BYTENUM[7:0] is written to a non-zero value.<br>0: When RELOAD=1, transfer of BYTENUM[7:0] bytes is not completed<br>1: When RELOAD=1, transfer of BYTENUM[7:0] bytes is completed |
| 6 | TC | Transfer complete in master mode<br>This bit is set by hardware when RELOAD=0, AUTOEND=0 and data of BYTENUM[7:0] bytes have been transferred. It is cleared by software when START bit or STOP bit is set.<br>0: Transfer of BYTENUM[7:0] bytes is not completed<br>1: Transfer of BYTENUM[7:0] bytes is completed |
| 5 | STPDET | STOP signal detected in slave mode<br>This flag is set by hardware when a STOP signal is detected on the bus. It is cleared by software by setting STPDETC bit and cleared by hardware when I2CEN=0.<br>0: STOP signal is not detected.<br>1: STOP signal is detected. |
| 4 | NACK | Not Acknowledge flag<br>This flag is set by hardware when a NACK is received. It is cleared by software by setting NACKC bit and cleared by hardware when I2CEN=0.<br>0: ACK is received.<br>1: NACK is received. |
| 3 | ADDSEND | Address received matches in slave mode.<br>This bit is set by hardware when the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDSENDC bit and cleared by hardware when I2CEN=0.<br>0: Received address not matched<br>1: Received address matched |
| 2 | RBNE | I2C_RDATA is not empty during receiving<br>This bit is set by hardware when the received data is shift into the I2C_RDATA register. It is cleared when I2C_RDATA is read.<br>0: I2C_RDATA is empty<br>1: I2C_RDATA is not empty, software can read |
| 1 | TI | Transmit interrupt<br>This bit is set by hardware when the I2C_TDATA register is empty and the I2C is ready to transmit data. It is cleared when the next data to be sent is written in the I2C_TDATA register.When SS=1, this bit can be set by software, in order to |

| | | |
|---|---|---|
| | | generate a TI event (interrupt if TIE=1 or DMA request if DENT =1).<br>0: I2C_TDATA is not empty or the I2C is not ready to transmit data<br>1: I2C_TDATA is empty and the I2C is ready to transmit data |
| 0 | TBE | I2C_TDATA is empty during transmitting<br>This bit is set by hardware when the I2C_TDATA register is empty. It is cleared when the next data to be sent is written in the I2C_TDATA register. This bit can be set by software in order to empty the I2C_TDATA register.<br>0: I2C_TDATA is not empty<br>1: I2C_TDATA is empty |

## 26.4.8. Status clear register (I2C_STATC)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SMBALTC | TIMEOUTC | PECERRC | OUERRC | LOSTARBC | BERRC | Reserved | | STPDETC | NACKC | ADDSENDC | Reserved | | |
| | | w | w | w | w | w | w | | | w | w | w | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | SMBALTC | SMBus alert flag clear.<br>Software can clear the SMBALT bit of I2C_STAT by writing 1 to this bit. |
| 12 | TIMEOUTC | TIMEOUT flag clear.<br>Software can clear the TIMEOUT bit of I2C_STAT by writing 1 to this bit. |
| 11 | PECERRC | PEC error flag clear.<br>Software can clear the PECERR bit of I2C_STAT by writing 1 to this bit. |
| 10 | OUERRC | Overrun/Underrun flag clear.<br>Software can clear the OUERR bit of I2C_STAT by writing 1 to this bit. |
| 9 | LOSTARBC | Arbitration Lost flag clear.<br>Software can clear the LOSTARB bit of I2C_STAT by writing 1 to this bit. |
| 8 | BERRC | Bus error flag clear.<br>Software can clear the BERR bit of I2C_STAT by writing 1 to this bit. |
| 7:6 | Reservced | Must be kept at reset value. |
| 5 | STPDETC | STPDET flag clear |

| | | |
|---|---|---|
| | | Software can clear the STPDET bit of I2C_STAT by writing 1 to this bit. |
| 4 | NACKC | Not Acknowledge flag clear<br>Software can clear the NACK bit of I2C_STAT by writing 1 to this bit. |
| 3 | ADDSENDC | ADDSEND flag clear<br>Software can clear the ADDSEND bit of I2C_STAT by writing 1 to this bit. |
| 2:0 | Reserved | Must be kept at reset value. |

### 26.4.9. PEC register (I2C_PEC)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | PECV[7:0] | | | | | | | |
| | | | | | | | | r | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | PECV[7:0] | Packet Error Checking Value that calculated by hardware when PEC is enabled.<br>PECV is cleared by hardware when I2CEN = 0. |

### 26.4.10. Receive data register (I2C_RDATA)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | RDATA[7:0] | | | | | | | |
| | | | | | | | | r | | | | | | | |

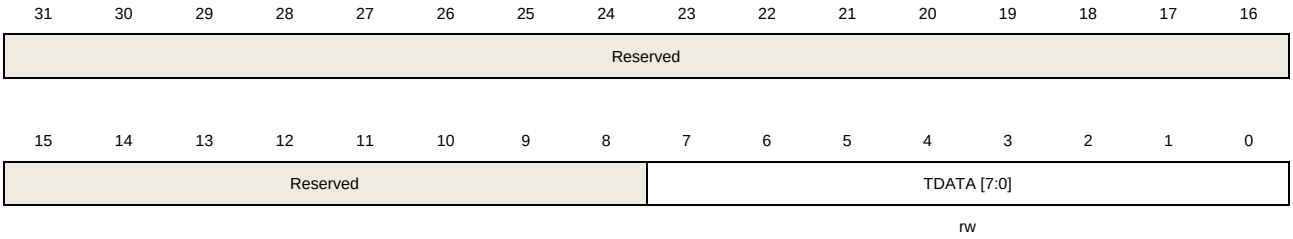
| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | RDATA[7:0] | Receive data value |

### 26.4.11. Transmit data register (I2C_TDATA)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



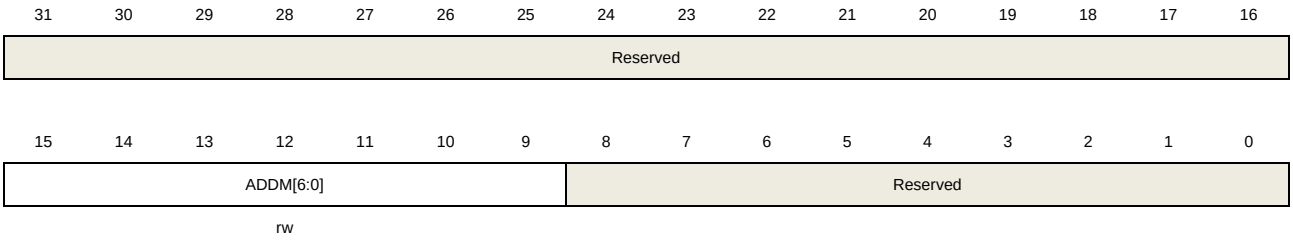

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | TDATA[7:0] | Transmit data value |

### 26.4.12. Control register 2 (I2C_CTL2)

Address offset: 0x90

Reset value: 0x0000 FE00

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:9 | ADDM[6:0] | Defines which bits of ADDRESS[7:1] are compared with an incoming address byte, and which bits are ignored. Any bit set to 1 in ADDM[6:0] enables comparisons with the corresponding bit in ADDRESS[7:1]. Bits set to 0 are ignored (can be either 0 or 1 in the incoming address). |
| 8:0 | Reserved | Must be kept at reset value. |

# 27. Serial peripheral interface/Inter-IC sound (SPI/I2S)

## 27.1. Overview

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol.

The serial peripheral interface (SPI) provides a SPI protocol of data transmission and reception function in master or slave mode. Both full-duplex, half-duplex and simplex communication modes are supported, with hardware CRC calculation and checking. Quad-SPI master mode is also supported in SPI3 / 4.

The inter-IC sound (I2S) supports four audio standards: I2S Philips standard, MSB justified standard, LSB justified standard, and PCM standard. I2S works at either master or slave mode for transmission and reception.

## 27.2. Characteristics

### 27.2.1. SPI characteristics

- Master or slave operation with full-duplex, half-duplex or simplex mode.
- Separate transmit and receive 32-bit FIFO.
- Data frame size can be 4 to 32 bits.
- Bit order can be LSB or MSB.
- Software and hardware NSS management, MOSI and MISO pin switch alternate function.
- Hardware CRC calculation, transmission and checking.
- Transmission and reception using DMA.
- SPI TI mode supported.
- Multi-master or multi-slave mode function.
- Protect configurations and settings.
- Both the minimum delay between data frames and the minimum delay between NSS and data stream are adjustable.
- Master mode failures can trigger interrupts, overrun or underrun flags, and CRC error detection.
- Adjustable main device receiver sampling time.
- Configurable FIFO thresholds (data packing).
- In slave mode, the underrun condition is configurable.
- Quad-SPI configuration available in master mode (in SPI3 / 4).

### 27.2.2. I2S characteristics

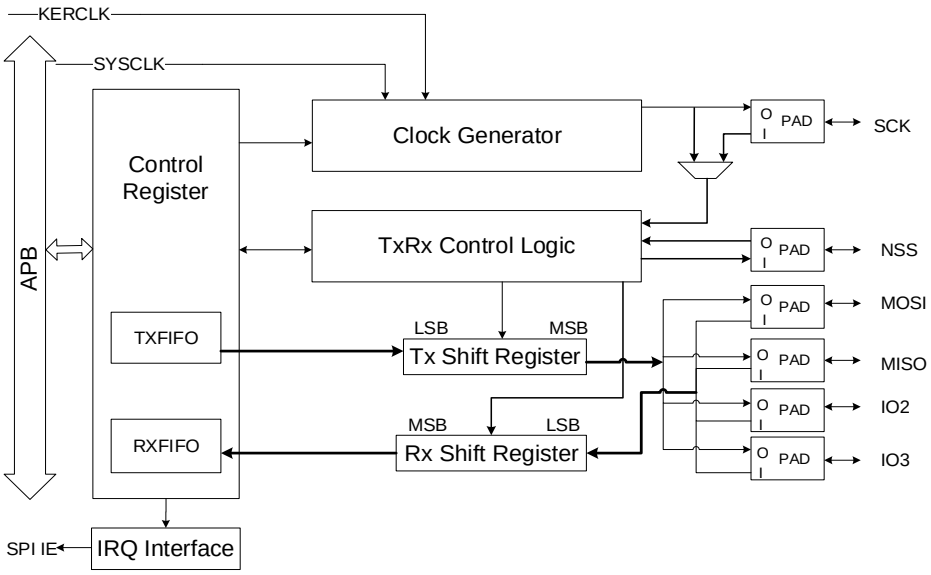
- Master or slave operation for transmission/reception.
- Four I2S standards supported: Philips, MSB justified, LSB justified and PCM standard.

- Data length can be 16 bits, 24 bits or 32 bits.
- Channel length can be 16 bits or 32 bits.
- Error signals improve reliability: underrun, overrun, and frame errors.
- Transmission and reception use a 32 bits wide buffer.
- Audio sample frequency can be 8 kHz to 192 kHz using I2S clock divider.
- Programmable idle state clock polarity.
- Master clock (MCK) can be output.
- Transmission and reception support DMA function.
- Separate transmit and receive 32-bit FIFO.

## 27.3. SPI function overview

### 27.3.1. SPI block diagram

**Figure 27-1. Block diagram of SPI**




- SYSCLK: system clock, provided by APB bus.
- KERCLK: kernel clock, provided by RCU. It has an asynchronous relationship with the system clock.
- The clock signal frequency must be consistent with user conditions and data transmission speed to prevent data loss. (**NOTE**: suggest the frequency of SYSCLK is greater than or equal to KERCLK)
- The SCK signal of the SPI slave is provided by the SPI master.

### 27.3.2. SPI signal description

#### Normal configuration (Not Quad-SPI Mode)

**Table 27-1. SPI signal description**

| Pin name | Direction | Description |
|---|---|---|
| SCK | I/O | Master: SPI clock output<br>Slave: SPI clock input |
| MISO | I/O | Master: Data reception line<br>Slave: Data transmission line<br>Master with bidirectional mode: Not used<br>Slave with bidirectional mode: Data transmission and reception line. |
| MOSI | I/O | Master: Data transmission line<br>Slave: Data reception line<br>Master with bidirectional mode: Data transmission and reception line.<br>Slave with bidirectional mode: Not used |
| NSS | I/O | Software NSS mode: not used<br>Master in hardware NSS mode: when NSSDRV = 1, it is NSS output, suitable for single master application; when NSSDRV = 0, it is NSS input, suitable for multi-master application.<br>Slave in hardware NSS mode: NSS input, as a chip select signal for slave. |

#### Quad-SPI configuration

SPI is in single wire mode by default and enters into Quad-SPI mode after QMOD bit in SPI_QCTL register is set (available in SPI3 / 4). Quad-SPI mode can only work in master mode.

The SPI is connected to external devices through 6 pins in Quad-SPI mode:

**Table 27-2. Quad-SPI signal description**

| Pin name | Direction | Description |
|---|---|---|
| SCK | O | SPI clock output |
| MOSI | I/O | Transmission / Reception data 0 |
| MISO | I/O | Transmission / Reception data 1 |
| IO2 | I/O | Transmission / Reception data 2 |
| IO3 | I/O | Transmission / Reception data 3 |
| NSS | O | NSS output |

#### Serial data line switching configuration

SPI can exchange the functions of MOSI and MISO pins through the SWPMIO bit in the SPI_CFG1 register.

**Table 27-3. MISO / MISO signal switching description**

| MODE | SWPMIO | MOSI | MISO |
|---|---|---|---|
| Master transmit | 0 | Transmission | - |
| | 1 | - | Transmission |
| Slave transmit | 0 | - | Transmission |
| | 1 | Transmission | - |
| Master receive | 0 | - | Reception |
| | 1 | Reception | - |
| Slave receive | 0 | Reception | - |
| | 1 | - | Reception |
| Master Full-duplex | 0 | Transmission | Reception |
| | 1 | Reception | Transmission |
| Slave Full-duplex | 0 | Reception | Transmission |
| | 1 | Transmission | Reception |

### 27.3.3. SPI clock timing and data format

CKPL and CKPH bits in SPI_CFG1 register decide the timing of SPI clock and data signal. The CKPL bit decides the SCK level when idle and CKPH bit decides either first or second clock edge is a valid sampling edge. These bits take no effect in TI mode.

**Figure 27-2. SPI timing diagram in normal mode**

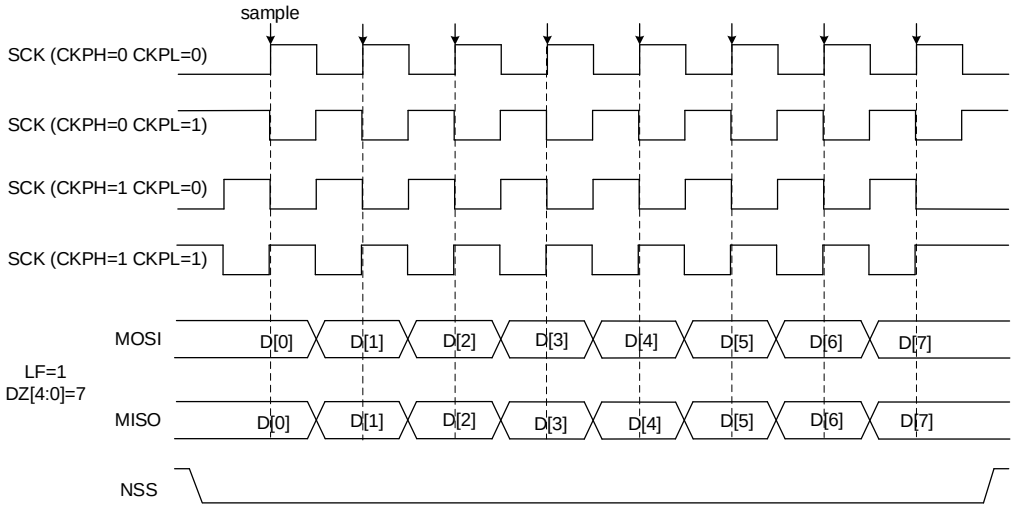
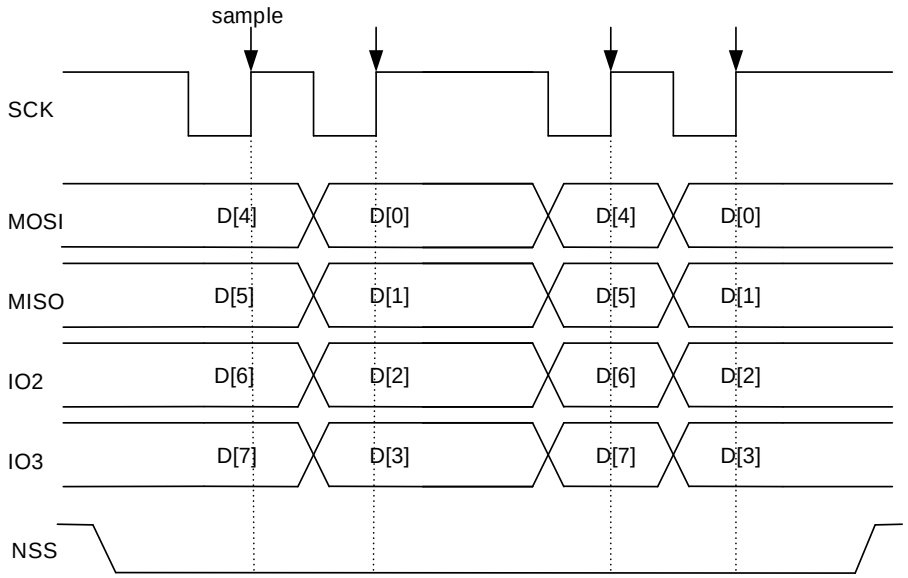


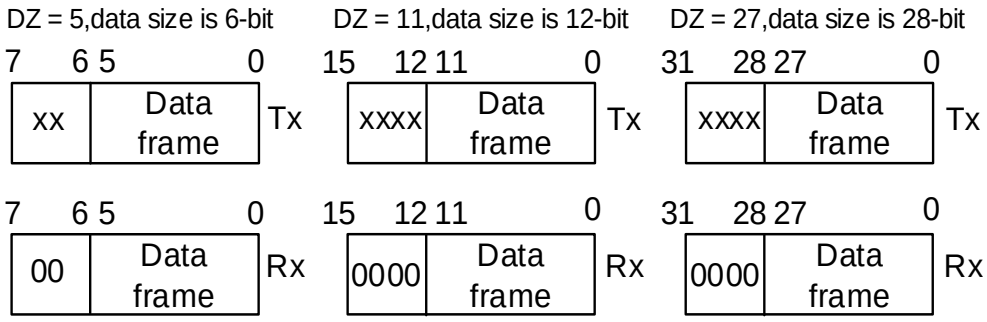
**Figure 27-3. SPI3 / 4 timing diagram in Quad-SPI mode (CKPL = 1, CKPH = 1, LF = 0)**




In SPI normal mode, the length of data is configured by the DZ[4:0] bits in the SPI_CFG0 register. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. Data order is configured by LF bit in SPI_CFG1 register, and SPI will first send the LSB if LF = 1, or the MSB if LF = 0. The data order is fixed to MSB first in TI mode.The data frame length is fixed to 8 bits in Quad-SPI mode.

When the SPI_TDATA / SPI_RDATA register is accessed, data frames are always right-aligned into either a byte (if the data fits into a byte), a half-word or a word. During communication, only bits within the data frame are clocked and transferred.

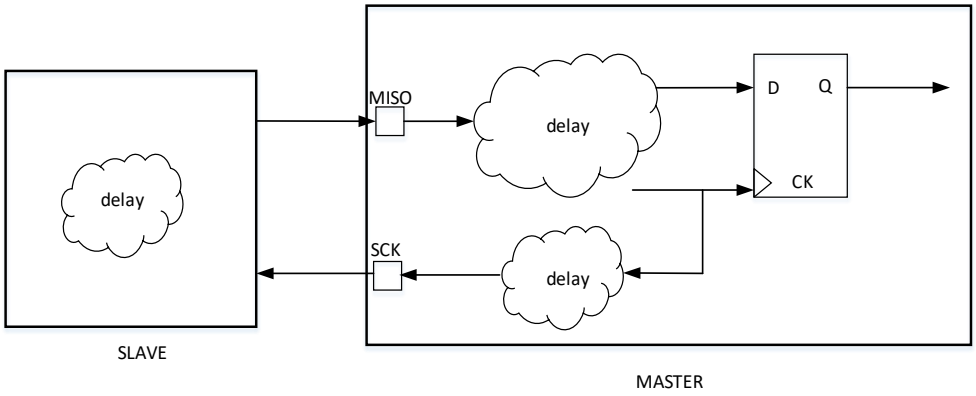
**Figure 27-4. SPI data frame right-aligned diagram**




## 27.3.4. SPI clock delay mode

SPI can be configured as master or slave mode. When SPI is configured as master, the SCK signal is sent to slave after delay. The slave send data by MISO, the data is sent to master sampling side after delay. Because of a series of delays will lead to the received data and clock have phase difference, resulting in data sampling error, it will be more obvious under high speed transmission.

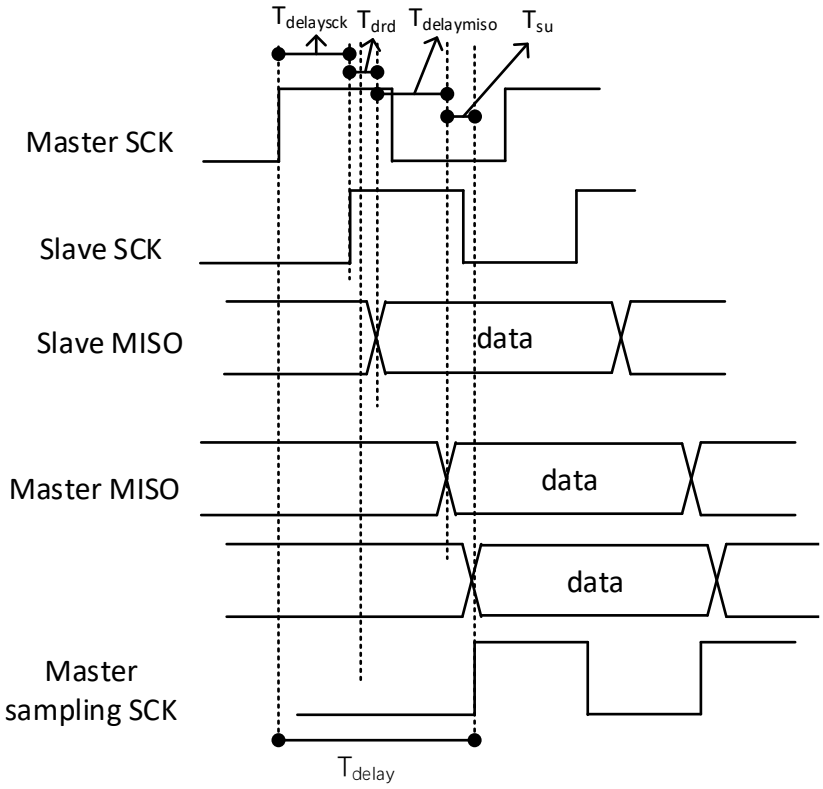
**Figure 27-5. SPI data and clock transmission path diagram**




In order to solve this problem, can be configured SPI_RXDLYCK register to adjust SPI internal receive clock phase that meet the correct sampling sequence(this configuration is optional, requires a combination of the actual scenario).

If the MRXDEN bit in the SPI_RXDLYCK register is set to 0, the delay function is opened, otherwise the delay function is closed. The MRXD[4:0] bits is used to configure delay units, the delay length can be configured to 1 ~ 32 units(one delay unit is 0.5ns in room temperature). User should configure $T_{delay}$ latency according to their own scenario($T_{delay} > T_{delaysck} + T_{drd} + T_{delaymiso} + T_{su}$).

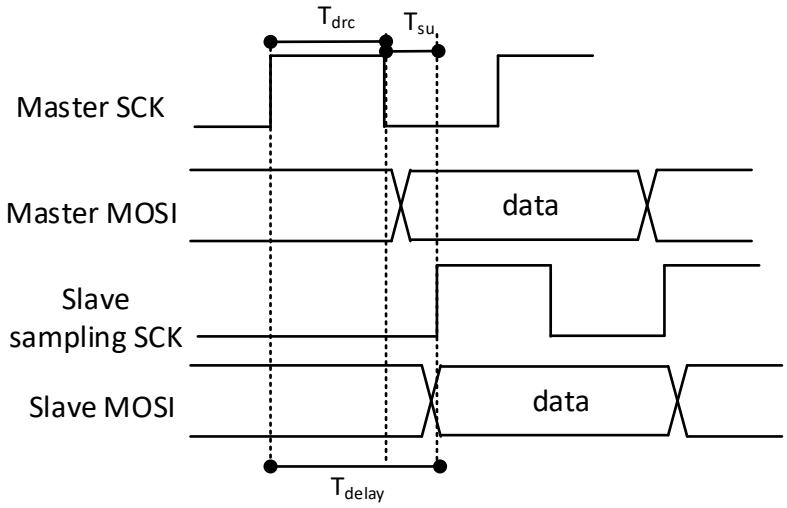
**Figure 27-6. SPI master rx delay configuration diagram**




If the SPI is configured as slave mode. If the SRXDEN bit in the SPI_RXDLYCK register is set to 0, the delay function is opened, otherwise the delay function is closed. The SRXD[4:0] bits is used to configure delay units, the delay length can be configured to 1 ~ 32 units(one

delay unit is 0.5ns in room temperature). User should configure $T_{delay}$ latency according to their own scenario($T_{delay} > T_{drc} + T_{su}$).

**Figure 27-7. SPI slave rx delay configuration diagram**




### 27.3.5. RxFIFO and TxFIFO

RxFIFO and TxFIFO are used in different directions for SPI data transactions, and they can enable the SPI to work in a continuous flow, and can prevent short data frame length or interrupt / DMA overrun occurs when the delay is too long.

A write access to the SPI_TDATA register stores the written data in the end of TxFIFO, while a read access to the SPI_RDATA returns the oldest value in RxFIFO which has not been read. FIFO processing depends on the data exchange mode (duplex and simplex), data frame format (DZ value), the size of the access to the FIFO register (8, 16, or 32 bits) and how the data is organized in the packet. The size of TxFIFO / RxFIFO is 16 x 32 bits and the maximum access data frame length is 32 bits. According to different frame length, the maximum number of frames that can be stored in FIFO is described in

***Table 27-4. The maxinum number of data frame stored in SPIX* FIFO.** (N = FIFO size / 32 = 16 x 32 / 32 = 16)

**Table 27-4. The maxinum number of data frame stored in SPIX FIFO**

| Frame size（DZ） | DZ <= 8bit | 8bit < DZ <= 16bit | 16bit < DZ <= 24bit | DZ > 24bit |
|---|---|---|---|---|
| Frame numbers (BYTEN = 1, WORDEN = 0) | N | - | - | - |
| Frame numbers (BYTEN = 0, WORDEN = 0) | 2N | N | | |
| Frame numbers | 4N | 2N | N | N |

| | | | | |
|---|---|---|---|---|
| (WORDEN = 1) | | | | |

**NOTE:** Both RxFIFO and TxFIFO content is kept flushed when SPI is disable (SPIEN = 0).

### RxFIFO reception

A read access to SPI_RDATA is managed by the RP event. This is triggered when RxFIFO is not considered to be empty (at least a complete data packet in RxFIFO). When RP is cleared, the RxFIFO is considered to be empty (or the data packet in RxFIFO is incomplete). RP triggers interrupts at RPIE position 1, or DMA requests at DMAREN position 1.

### TxFIFO transmission

Write access to SPI_TDATA is managed by the TP event. This event is triggered when TxFIFO has enough available space to receive a packet. If the TxFIFO is filled by software or DMA, the TP flag cleared. If there is not enough space to store at least one data packet when TXF set to 1 or SPI disable, write to TxFIFO is will be ignored. TP triggers interrupts at TPIE position 1, or DMA requests at DMATEN position 1. TPIE shield is cleared by hardware when TXF flag is set to 1.

### Duplex packet handing

In full-duplex mode, the DP bit can monitor the TP and RP events. With the DP flag set to 1, the application writes an appropriate number of data to the SPI_TDATA register to upload one complete packet, and then reads an equal number of data from the SPI_RDATA register to download one complete packet. After a packet is uploaded and downloaded, the application checks the DP value to see if it can be ordered push in and pop up other packets, and if possible, upload/download them packet by packet until DP reads 0. DP triggers interrupts with DPIE set 1, or DMA requestion when both DMATEN and DMAREN set to 1. DPIE shield is cleared by hardware when TXF flag is set to 1.

If the next data is received while the RxFIFO is full, the reception overrun event occurs. Overrun events can be polled or handled by interrupts. This may occur in slave mode or master mode (full-duplex or receive-only mode, MASP = 0). The main device is in receive-only mode, when MASP = 1, if the RxFIFO is full, the generated clock will stop automatically to prevent the overrun events.

### Data packing

When the data frame size (DZ) <= 8 bits, data packing mode is automatically enabled when any 16-bit or 32-bit read or write access in SPI_RDATA or SPI_TDATA (BYTEN = 0 or WORDEN = 1). The multi-data frames pattern is handled in parallel in this case. At the sending terminal, if FIFOLVL = 1 (2 data frames in a packet) or FIFOLVL = 3 (4 data frams in a packet), two or four data frames are sent after the single 16-bit or 32-bit access the SPI_TDATA register of the transmitter. At the receiving terminal, if FIFOLVL = 1 (2 data frames in a packet) or FIFOLVL = 3 (4 data frams in a packet), two or four data frames are received simultaneously after the single 16-bit or 32-bit access the SPI_RDATA register of the receiver, which can generate just one RP event in the receiver. The receiver then has to

read all data frames by 16-bit or 32-bit from SPI_RDATA. If FIFOLVL = 0 (1 data frames in a packet), the receiver will generate two or four RP events when reading a data frame by 16-bit or 32-bit from SPI_RDATA.

If 9-bit <= DZ <= 16-bit, data packing mode is automatically enabled when any 32-bit read or write access in SPI_RDATA or SPI_TDATA. The least significant halfword will be used to store significant data. At the sending terminal, if FIFOLVL = 1 (2 data frames in a packet), two data frames are sent after the single 32-bit access the SPI_TDATA register of the transmitter. At the receiving terminal, if FIFOLVL = 1 (2 data frames in a packet), two data frames are received simultaneously after the single 32-bit access the SPI_RDATA register of the receiver, which can generate just one RP event in the receiver. The receiver then has to read all data frames by 32-bit from SPI_RDATA. If FIFOLVL = 0 (1 data frames in a packet), the receiver will generate two RP events when reading a data frame by 32-bit from SPI_RDATA.

When short data frame size (< 8-bit or < 16-bit) are paired with large data access patterns (16-bit or 32-bit), the FIFOLVL value must be configured as a multiple of the number of data frames , multiple of 4 if 32-bit access is used for frames up to 8 bits, multiple of 2 if 16-bit access is used for frames up to 8-bit or multiple of 2 while 32-bit access is used for frames up to 16-bit.

The FIFOLVL setting must always be higher than the subsequent read access size, otherwise additional pseudo-data will be read. FIFO data access that is smaller than the configured data size is not allowed (Data frame size is configured by DZ, the size of FIFO data access is configured by BYTEN and WORDEN). Always ensure that at least one complete data frame is accessed.

### Sequential transaction handing

Users can processes multiple numbers of data in a message according to TXSIZE and TXSER value. The transaction of a message starts when SPI is enabled by setting MSTART bit and finishes when the number of data required has been transferred. If TXSIZE remains zero when MSTART is set to 1, the infinite transaction is initialized. Transactions can be suspended at any time by setting the MSPDR bit, which clears MSTART bit.

In master mode, after the number of data in TXSIZE has been transferred, if TXSER contains a non-zero value, the value of TXSER will be copied to TXSIZE and the TXSER value is cleared automatically. The transmission will then increase the number of data corresponding to the newly loaded value in TXSIZE. After the reload operation, if TXSERFIE is set to 1, the TXSERF flag is set to 1 and will triggers interrupt. The user can write the next non-zero value to the TXSER before the next reload, so it can handle multiple data. In this case, ET events do not occur because transmission continues.

If data amount of TXSIZE or TXSER (number of data frames) is not aligned with packet length defined in FIFOLVL, then the last incomplete packet before the end of sending needs to be packaged. ***Data packing*** detail describe the principle of packaging.

**Note**: in order to prevent transmission underrun, the SPI_URDATA register of slave can be written to a specific value. When the TxFIFO of slave becomes empty, the value in its

SPI_URDATA will be the next data is sent out automatically, and after the host receives this value which can be parsed by the software, so the host can suspend its receiver through software operation.

#### Transaction delay handing

If the reception speed of slave is less than the transmission speed of master, the master must be to reduce the transmission speed, by lowering the clock frequency or increase the time delay between data frames. The MDFD[3:0] bits in SPI_CFG1 register is used to add delay between data frames in master mode, and MSSD[3:0] is used to add delay between active edge of NSS and start transfer or receive data in master mode. ***NSS signal timing*** shows the detail description.

### 27.3.6. NSS function

#### Slave mode

When slave mode is configured (MSTMOD = 0), SPI gets NSS level from NSS pin in hardware NSS mode (NSSIM = 0) or from NSSI bit in software NSS mode (NSSIM = 1) and transmits / receives data only when NSS level is active. The user can set NSSIOPL bit to decide the active level of input/output external signals (on NSS pins).

**Table 27-5. NSS function in slave mode**

| Mode | Register configuration | Description |
|---|---|---|
| Slave hardware NSS mode | MSTMOD = 0<br>NSSIM = 0 | SPI slave gets NSS level from NSS pin. |
| Slave software NSS mode | MSTMOD = 0<br>NSSIM = 1 | SPI slave NSS level is determined by the NSSI bit.<br>NSSI = 0: NSS level is low<br>NSSI = 1: NSS level is high |

#### Master mode

In master mode (MSTMOD = 1) if the application uses multi-master connection, NSS can be configured to hardware input mode (NSSIM = 0, NSSDRV = 0) or software mode (NSSIM = 1). Then, once the NSS pin (in hardware NSS mode) or the NSSI bit (in software NSS mode) goes non-active, the SPI automatically enters slave mode and triggers a master fault flag CONFERR.

If the application wants to use NSS line to control the SPI slave, NSS should be configured to hardware output mode (NSSIM = 0, NSSDRV = 1). After enabling SPI and setting MSTART, NSS remains active until the current amount of data to be transmitted is completed (ET flag is set), at which point NSS becomes inactive.

Applications can use a generic I/O port as an NSS pin for more flexible NSS applications.

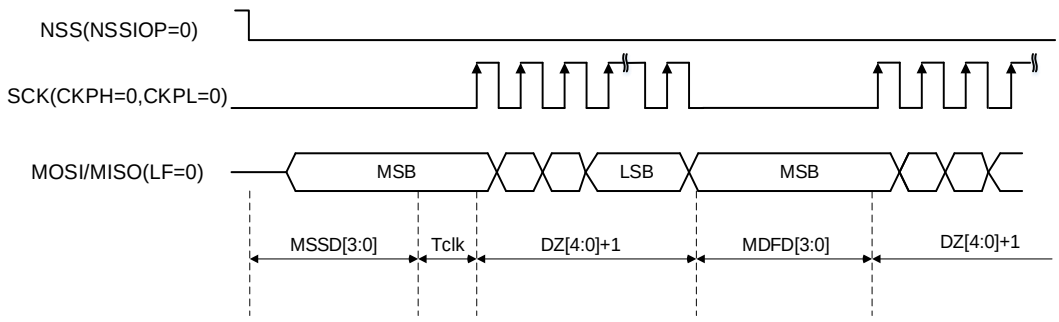
**Table 27-6. NSS function in master mode**

| Mode | Register configuration | Description |
|---|---|---|
| Master hardware NSS output mode | MSTMOD = 1<br>NSSIM = 0<br>NSSDRV = 1 | Applicable to single-master mode. The master uses the NSS pin to control the SPI slave device. At this time, the NSS is configured as the hardware output mode. NSS become active level after enabling SPI and setting MSTART. |
| Master hardware NSS input mode | MSTMOD = 1<br>NSSIM = 0<br>NSSDRV = 0 | Applicable to multi-master mode. At this time, NSS is configured as hardware input mode. Once the NSS pin become non-active level, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be set to 1. |
| Master software NSS mode | MSTMOD = 1<br>NSSIM = 1<br>NSSI = 0<br>NSSDRV: Don't care | Applicable to multi-master mode. Once NSS become non-active level, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be 1. |
| | MSTMOD = 1<br>NSSIM = 1<br>NSSI = 1<br>NSSDRV: Don't care | The slave can use hardware or software NSS mode. |

### NSS signal timing

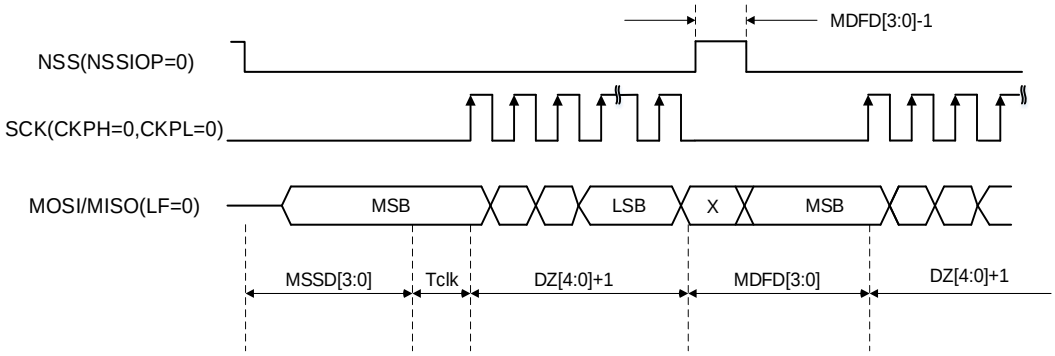
When applying hardware output NSS control (NSSIM = 0, NSSDRV = 1), users can configure MDFD[3:0] and MSSD[3:0] bit fields to control NSS signal timing between data frames and insert additional delay at the start of each transaction (to separate NSS and clock start). *Figure 27-8. NSS signal delay timing diagram (MSSD[3:0] = 0011 (3 x $T_{clk}$), MDFD = 0011 (3 x $T_{clk}$))* shows the effective delay time between data singal and NSS signal with MSSD[3:0] = 3, and the effective delay time between data frames with MDFD[3:0] = 3.

**Figure 27-8. NSS signal delay timing diagram (MSSD[3:0] = 0011 (3 x $T_{clk}$), MDFD = 0011 (3 x $T_{clk}$))**



When NSSCTL = 1 and MDFD[3:0] > 1, interlaced pulses can be inserted between SPI data frames. ***Figure 27-9. NSS interlaced pulses timing diagram (MSSD[3:0] = 0011 (3 x $T_{clk}$), MDFD = 0011 (3 x $T_{clk}$))*** shows the NSS signal pluse status with MDFD[3:0] > 1.

**Figure 27-9. NSS interlaced pulses timing diagram (MSSD[3:0] = 0011 (3 x $T_{clk}$), MDFD = 0011 (3 x $T_{clk}$))**




## 27.3.7. SPI operation modes

**Table 27-7. SPI operation modes**

| Mode | Description | Register configuration | Data pin usage |
|---|---|---|---|
| MFD | Master full-duplex | MSTMOD = 1<br>RO = 0<br>BDEN = 0<br>BDOEN: Don't care | MOSI: Transmission<br>MISO: Reception |
| MTU | Master transmission with unidirectional connection | MSTMOD = 1<br>RO = 0<br>BDEN = 0<br>BDOEN: Don't care | MOSI: Transmission<br>MISO: Not used |
| MRU | Master reception with unidirectional connection | MSTMOD = 1<br>RO = 1<br>BDEN = 0<br>BDOEN: Don't care | MOSI: Not used<br>MISO: Reception |
| MTB | Master transmission with bidirectional connection | MSTMOD = 1<br>RO = 0<br>BDEN = 1<br>BDOEN = 1 | MOSI: Transmission<br>MISO: Not used |
| MRB | Master reception with bidirectional connection | MSTMOD = 1<br>RO = 0<br>BDEN = 1<br>BDOEN = 0 | MOSI: Reception<br>MISO: Not used |
| SFD | Slave full-duplex | MSTMOD = 0<br>RO = 0<br>BDEN = 0 | MOSI: Reception<br>MISO: Transmission |

| Mode | Description | Register configuration | Data pin usage |
|---|---|---|---|
| | | BDOEN: Don't care | |
| STU | Slave transmission with unidirectional connection | MSTMOD = 0<br>RO = 0<br>BDEN = 0<br>BDOEN: Don't care | MOSI: Not used<br>MISO: Transmission |
| SRU | Slave reception with unidirectional connection | MSTMOD = 0<br>RO = 1<br>BDEN = 0<br>BDOEN: Don't care | MOSI: Reception<br>MISO: Not used |
| STB | Slave transmission with bidirectional connection | MSTMOD = 0<br>RO = 0<br>BDEN = 1<br>BDOEN = 1 | MOSI: Not used<br>MISO: Transmission |
| SRB | Slave reception with bidirectional connection | MSTMOD = 0<br>RO = 0<br>BDEN = 1<br>BDOEN = 0 | MOSI: Not used<br>MISO: Reception |

**Figure 27-10. A typical full-duplex connection**

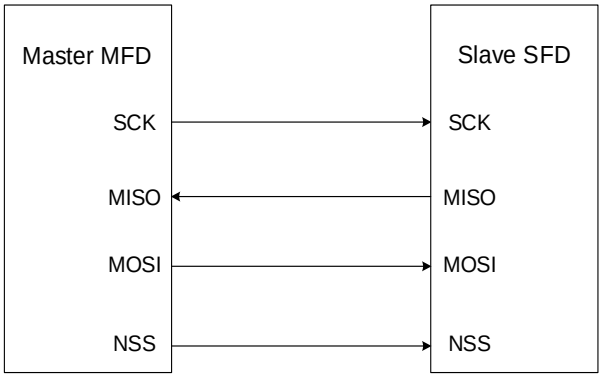



**Figure 27-11. A typical simplex connection (Master: Receive, Slave: Transmit)**

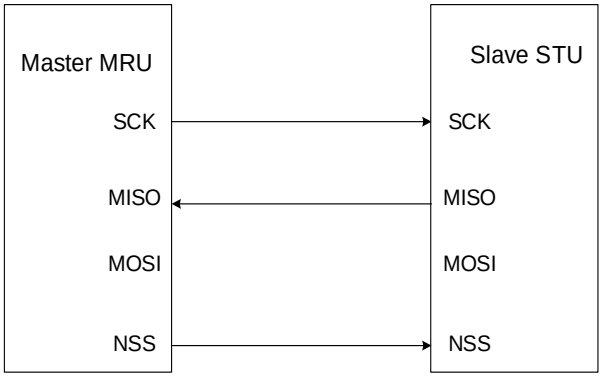


**Figure 27-12. A typical simplex connection (Master: Transmit only, Slave: Receive)**

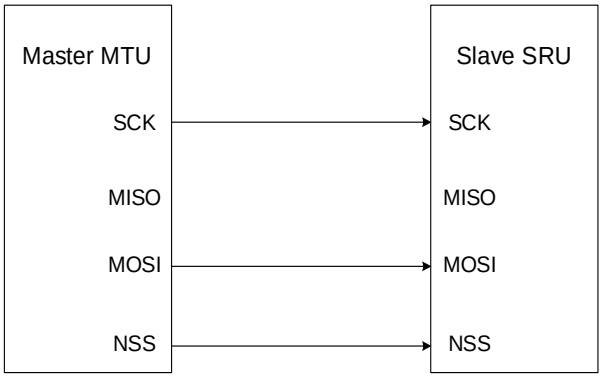
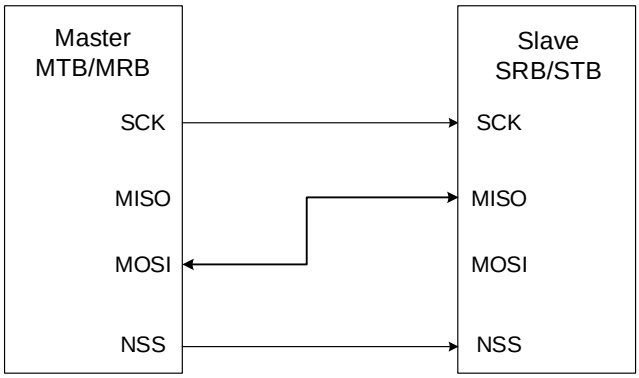



**Figure 27-13. A typical bidirectional connection**




### SPI initialization sequence

Before transmitting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode or TI mode is used, program the PSC [2:0] bits in SPI_CFG0 register to generate SCK with desired baud rate or configure the Td time in TI mode, otherwise, ignore this step.
2. Program the clock timing register (CKPL and CKPH bits in the SPI_CFG1 register).
3. Program the frame format (LF bit in the SPI_CFG1 register).
4. Program data format (DZ[4:0] bits in the SPI_CFG0 register).
5. Program the FIFO level (FIFOLVL[3:0] bits in the SPI_CFG0 register), and FIFO access size (WORDEN and BYTEN).
6. Program the NSS mode (NSSIM / NSSDRV / NSSIOPL / NSSCTL / MDFD[3:0] / MSSD[3:0] bits in the SPI_CFG1 register, NSSI bit in the SPI_CTL0 register) according to the application's demand as described above in ***NSS function*** section.
7. If slave mode, program the TXURDT[1:0] and TXUROP[1:0] bits in SPI_CFG0 register.
8. If TI mode is used, set TMOD bit in SPI_CFG1 register, otherwise, ignore this step.
9. Configure MSTMOD, RO, BDEN and BDOEN depending on the operating modes described in ***SPI operation modes*** section.
10. Program the SPI_CTL1 register to select the length of the transmission, and if the value is unknown, TXSIZE must be programmed to zero.

11. Program the SPI_CRCPOLY register, and program the CRCSZ[4:0] bit fields and CRCFS bit to configure CRC polynomials and CRC calculations as needed, described in ***CRC function*** section.
12. Initialize DMATEN / DMAREN bits if they are needed when DMA is used, described in ***DMA function*** section.
13. If Quad-SPI mode is used, set the QMOD bit in SPI_QCTL register. Ignore this step if Quad-SPI mode is not used.(only used in SPI3 / 4)
14. If you need to configure protection, program the IOAFEN bit of the SPI_CTL0 register.
15. Enable the SPI (set the SPIEN bit).
16. If master mode (MSTMOD =1), when SPIEN=1, program the MSTART bit in SPI_CTL0 to transfer data. Ignore this step if no data transfer is required.

**Note**: During communication, CKPH, CKPL, MSTMOD, PSC[2:0] and LF bits should not be changed.

## SPI basic transmission and reception sequence

### Transmission sequence

After the initialization sequence, the SPI is enabled and stays at idle state. In master mode, the transmission starts when the application writes a data into the TxFIFO. In slave mode the transmission starts when SCK clock signal begins to toggle at SCK pin and NSS level is active, so application should ensure that data is already written into TxFIFO before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the TxFIFO to the shift register and then begins to transmit the loaded data frame. The related operation is described in ***RxFIFO and TxFIFO*** sections.

Write access to SPI_TDATA is managed by the TP event. With the TP flag set to 1, the application performs an appropriate number of SPI data register writes to transfer the contents of the data package. After uploading the new complete package, the application checks the TP value to check whether TxFIFO can receive additional data packet, if TP = 1, they are uploaded packet by packet until TP reads 0. If the transmission size and packet size are not aligned, when the last number of data packets to be transferred cannot reach the configured size (set by FIFOLVL). The application can still write standard number of previous complete packets to TxFIFO without adverse effects: only consistent data (complete data frames) will pull down to TxFIFO, while redundant write times (or any incomplete data) will be ignored.

In master mode, software should write the next data into SPI_TDATA register before the transmission of current data frame is completed if it desires to generate continuous transmission. As long as there is data in TxFIFO, data delivery continues until TxFIFO becomes empty.

### Reception sequence

After the last valid sample clock, the incoming data will be moved from shift register to the RxFIFO and RP will be set to 1. The application should read SPI_RDATA register to get the

received data and this will clear the RP flag automatically when the number of data less than FIFOLVL in RxFIFO. In MRU and MRB modes, hardware continuously sends clock signal to receive the next data frame, while in full-duplex master mode (MFD), hardware only receives the next data frame when the TxFIFO is not empty. The related operation is described in ***RxFIFO and TxFIFO*** sections.

A read access to SPI_RDATA is managed by the RP event. With the RP flag set to 1, the application performs an appropriate number of SPI data register reads to download a single piece of data contents of the package. After the complete packet is downloaded, the application checks the RP value to see whether there are other packets in the RxFIFO, if any, they are downloaded packet by packet until RP reads 0. At the end of the receive, it may occur that some data is still available in the RxFIFO without reaching the FIFOLVL level, so RP will not be set to 1. In this case, the number of remaining RX data frames in the RxFIFO will be indicated by RWNE and RPLVL in the SPI_STAT register. If the transmission size and packet size are not aligned, the above condition occurs when the last number of data packets to be received cannot reach the configured size (set by FIFOLVL). However, the application can still read standard number of previous complete packets from RxFIFO without adverse effects: only consistent data (complete data frames) will pull up from RxFIFO, while redundant read times (or any incomplete data) will read 0.

When receiving data, the master provides clock signal, and the receiving process is stopped when the master stops or suspends the SPI interface. The master initiates the process by setting MSTART to 1, which can be suspended by writing 1 to the MSPDR of the SPI_CTL0 register, or by writing 1 to the MASP bit. The receiving process also complete when the data frames number in TXSIZE and TXSER are transferred.

### SPI operation sequence in different modes (Not Quad-SPI, TI mode)

In full-duplex mode, either MFD or SFD, the RP and TP flags should be monitored and then follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to the transmission sequence of full-duplex mode regardless of the RP and RXORERR bits.

The master reception mode (MRU or MRB) is different from the reception sequence of full-duplex mode. In MRU or MRB mode, after SPI is enabled, the SPI continuously generates SCK until the SPI is disabled. So the application should ignore the TP flag and read out RxFIFO in time after the RP flag is set, otherwise a data overrun fault will occur.

The slave reception mode (SRU or SRB) is similar to the reception sequence of full-duplex mode regardless of the TP flag.

### SPI TI mode

SPI TI mode takes NSS as a special frame header flag signal and its operation sequence is similar to normal mode described above. The modes described above (MFD, MTU, MRU, MTB, MRB, SFD, STU, SRU, STB and SRB) are still supported in TI mode. While, in TI mode the CKPL, CKPH, LF, NSSIM, NSSIOPL, NSSDRV bits take no effect and the SCK sample

edge is falling edge.

**Figure 27-14. Timing diagram of TI master mode with discontinuous transfer**

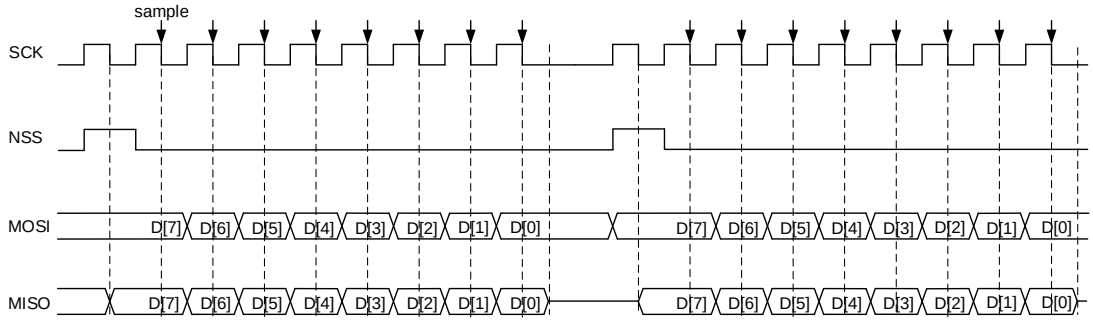
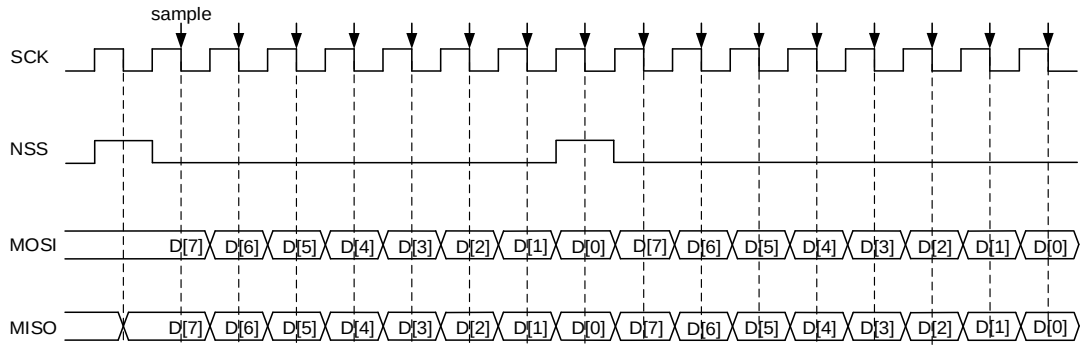
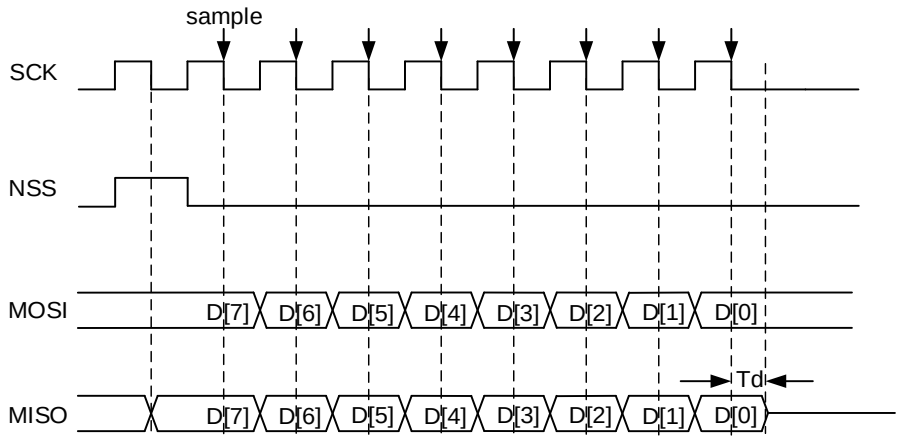



**Figure 27-15. Timing diagram of TI master mode with continuous transfer**




In master TI mode, SPI can perform continuous or non-continuous transfer. If the master writes SPI_TDATA register fast enough, the transfer is continuous, otherwise non-continuous. In non-continuous transfer there is an extra header clock cycle before each byte. While in continuous transfer, the extra header clock cycle only exists before the first byte and the following bytes' header clock is overlaid at the last bit of pervious bytes.

**Figure 27-16. Timing diagram of TI slave mode**




In slave TI mode, after the last rising edge of SCK in transfer, the slave begins to transmit the LSB bit of the last data byte, and after a half-bit time, the master begins to sample the line. To make sure that the master samples the right value, the slave should continue to drive this bit after the falling sample edge of SCK for a period of time before releasing the MISO pin. This time is called $T_d$. $T_d$ is decided by PSC [2:0] bits in SPI_CFG0 register.

$$\frac{T_{bit}}{2} + 2 * T_{kerclk} \leq T_d \leq \frac{T_{bit}}{2} + 4 * T_{kerclk} \quad (27\text{-}1)$$

In slave mode, the slave also monitors the NSS signal and sets an error flag FERR if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte.

### Quad-SPI mode operation sequence

The Quad-SPI mode is designed to control Quad-SPI flash.

In order to enter Quad-SPI mode, the software should first verify that the TP and TC bits is set, then set QMOD bit in SPI_QCTL register. In Quad-SPI mode, BDEN, BDOEN, CRCEN, CRCSZ, RO and LF bits should be kept cleared and DZ[4:0] should be set to ensure that SPI data size is 8-bit, MSTMOD should be set to 1 for SPI is used in master mode. WORDEN should be set to 1. SPIEN, MSTART, TXSIZE, TXSER, PSC, CKPL and CKPH should be configured as desired.

**Note:** The CRC function is not supported in Quad-SPI mode. The PSC cannot be configured with two or four division frequencies.

There are two operation modes in Quad-SPI mode: quad write and quad read, decided by QRD bit in SPI_QCTL register.

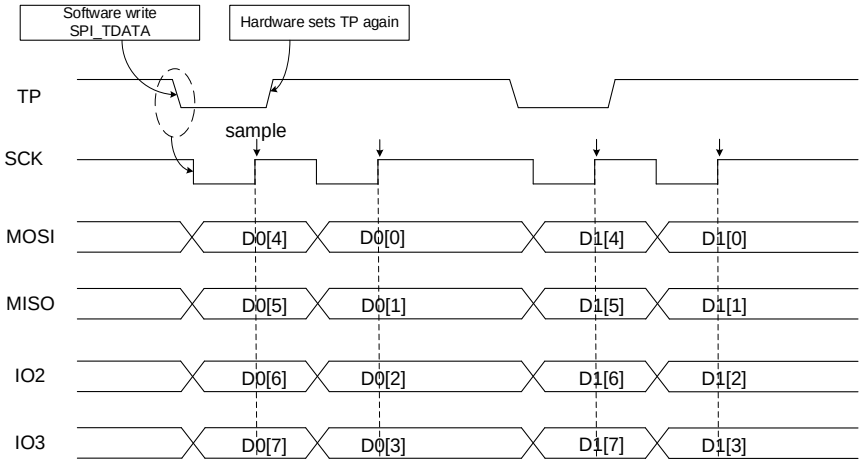
#### Quad write operation

SPI works in quad write mode when QMOD is set and QRD is cleared in SPI_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as output pins. SPI begins to generate clock on SCK line and transmit data on MOSI, MISO, IO2 and IO3 as soon as data is written into SPI_TDATA (TP is cleared) with SPIEN and MSTART bits are set. Once SPI starts transmission, it always checks TP status at the end of a frame and stops when condition is not met.

The operation flow for transmitting in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 / SPI_CTL1 / SPI_CFG0 / SPI_CFG1 based on your application requirements.
2. Set QMOD bit in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0 register.
3. Write a byte to SPI_TDATA register and the TP will be cleared.
4. Wait until TP is set by hardware again before writing the next byte.

**Figure 27-17. Timing diagram of quad write operation in Quad-SPI mode**



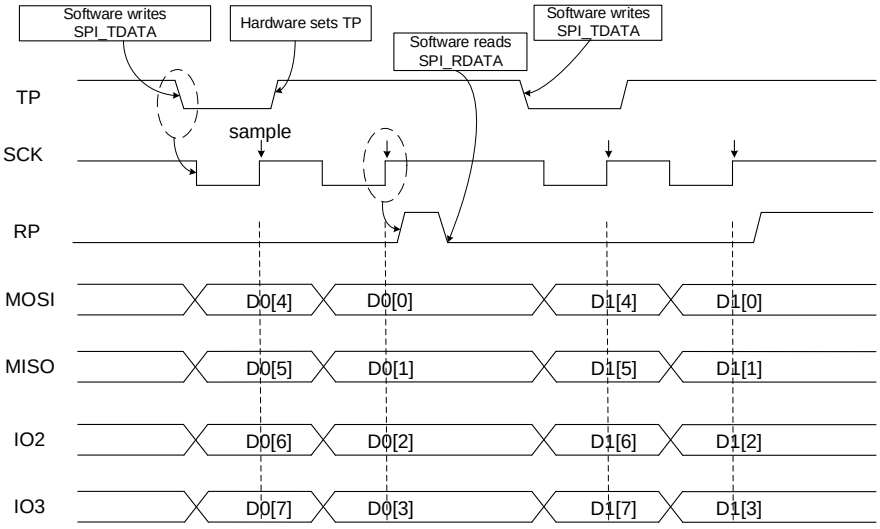

### Quad read operation

SPI works in quad read mode when QMOD and QRD are both set in SPI_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as input pins. SPI begins to generate clock on SCK line as soon as a data is written into SPI_TDATA (TP is cleared) and SPIEN is set. Writing data into SPI_TDATA is only to generate SCK clocks, so the written data can be any value. Once SPI starts transmission, it always checks SPIEN and TP status at the end of a frame and stops when condition is not met. So, dummy data should always be written into SPI_TDATA to generate SCK.

The operation flow for receiving in quad mode is shown below:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 / SPI_CTL1 / SPI_CFG0 / SPI_CFG1 register based on your application requirements.
2. Set QMOD and QRD bits in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0 register.
3. Write an arbitrary byte (for example, 0xFF) to SPI_TDATA register.
4. Wait until the RP flag is set and read SPI_RDATA to get the received byte.
5. Write an arbitrary byte (for example, 0xFF) to SPI_TDATA to receive the next byte.

**Figure 27-18. Timing diagram of quad read operation in Quad-SPI mode**




## SPI disabling sequence

Different sequences are used to disable the SPI in different operation modes:

### MFD MTU MTB SFD STU STB

Any transaction can be terminated when the device in full duplex or send only mode which stops providing data to be sent. In this In this case, the clock stops after the last data has been transferred. The TC flag can be polled (or enable interrupt through ESTCIE = 1) to wait for the last data frame to be sent. Wait for TC = 1 or ET = 1 (no more data to be sent and send the last data frame). If enable CRC function, CRC will be sent automatically after the last data is processed. In this case TC/ET will be set to 1 after CRC frame completion. When sending is suspended, the software must wait until the MSTART bit is cleared. Then disable the SPI by clearing SPIEN bit.

### MRU MRB

To stop peripherals, SPI communication must first be suspended by setting MSPDR to 1 or wait for data transmission complete according to ET flag when the master device is in receive-only mode. If the receive flow is paused, please wait for SPD = 1. When the SPI is suspended, the data received but not read is always stored in the RxFIFO (When SPI is disable, RxFIFO will be cleared). Read all RxFIFO data (until RWNE = 0 and RPLVL = 0). Then disable the SPI by clearing SPIEN bit.

### SRU SRB

Application can disable the SPI when it doesn't want to receive data, and any ongoing data will be lost.

**TI mode**

The disabling sequence of TI mode is the same as the sequences described above.

**Quad-SPI mode**

Application can operate as MFD mode, then the QMOD bit in SPI_QCTL register and SPIEN bit in SPI_CTL0 register are cleared.

### 27.3.8. DMA function

The DMA frees the application from data writing and reading process during transfer, to improve the system efficiency.

DMA function in SPI is enabled by setting DMATEN and DMAREN bits in SPI_CFG0 register. To use DMA function, application should first correctly configure DMA modules, then configure SPI module according to the initialization sequence, at last enable SPI.

After initialization finish, If DMATEN is set, SPI will generate a DMA request each time when TP = 1, then DMA will acknowledge to this request and write data into the SPI_TDATA register automatically. If the data to transfer is not ready, TP and TXURERR will be set to 1. In this case, data will be sent based on the TXUROP bit selection. If DMAREN is set, SPI will generate a DMA request each time when RP = 1, then DMA will acknowledge to this request and read data from the SPI_RDATA register automatically. If the ET is set to 1 at the end of the transaction and the last packet is incomplete, DMA requests are automatically activated to read the rest of the data according to RWNE and RPLVL[1:0] settings(in SPI_STAT register).

**Data packing with DMA**

If the transfers are managed by DMA (DMATEN = 1 or DMAREN = 1), when DZ[4:0] <= 8-bit, and SPI_TDATA register is accessed in 16-bits or 32-bit, or 8-bit < DZ[4:0] <= 16-bit , and SPI_TDATA register is accessed in 32-bit, the DMA data packing mode is enabled, the DMA should automatically manages the write operations to the SPI_TDATA register.

Regardless of the data packing mode used, and regardless of whether the number of data to be transferred is multiple of DMA data size (16 or 32 bits). When the frame size is small, DMA will automatically complete the transfer based on the TXSIZE field setting. To configure DMA, DMA data access that is smaller than the configured data size is forbidden. Always be sure the data to access at least one complete data frames.

### 27.3.9. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial defined in SPI_CRCPOLY register. The polynomial length is defined by the most significant bit of SPI_CECPOLY register. If DZ <= 32-bit, CRC polynomial length supports 5 – 33 bits. If DZ <= 16-bit, CRC polynomial length supports 5 – 17 bits. The CRC polynomial length must be greater than the DZ value. If DZ = 32-bit, the

CRCFS bit in SPI_CTL0 register must be set to 1 to make sure CRC polynomial used in full scale mode. The CRCSZ bit fields in SPI_CFG0 register can define the most significant bit number processed by the CRC calculator and comparing with CRC frame.

Application can enable the CRC function by setting CRCEN bit in SPI_CFG0 register, it must be configured before SPI is enabled. The CRCPOLY value can only be modified when there is no data transmission, and it is recommended to configure it when CRCEN=0. The CRC calculators continuously calculate CRC for each bit transmitted and received on lines, and the calculated CRC values can be read from SPI_TCRC and SPI_RCRC registers. Sending and receiving of CRC value is implemented in the form of frame, the frame length is equal to the length of the CRCSZ bit fields in SPI_CTL0 register.

In transmission phase, hardware send the calculated CRC value stored in SPI_TCRC register automatically after the last data is written to the TxFIFO. In reception phase, the CRC value is stored into SPI_RCRC register after the last data is read from the RxFIFO. The CRC calculator will get CRC value by performing CRC calculation to the received data, and the CRC value will be compare with the SPI_RCRC value. When CRC checking fails, the CRCERR flag will be set. Writing 1 to CRCERRC bit in SPI_STATC register can clear the CRCERR bit.

**Note**: when the SPI is disable, or at the beginning of the new data sampling after the last data transmission is completed, the SPI_TCRC and SPI_RCRC registers will be initialized, the value of the initialization can be defined by TXCRCI and RXCRCI bit in SPI_CTL0 register.

When SPI is configured as slave mode and the CRC function is used, the CRC calculation will still be performed even when the NSS pin is high (when the NSS signal is high, as long as there is a clock pulse on the SCK pin, the CRC calculation will continue).This situation occurs when the master device alternately communicates with multiple slave devices. At this time, it is recommended to restart the CRC function when the NSS signal is low.When the slave device is not selected (NSS signal is high) and switches to being selected as a new slave device (NSS signal is low), in order to maintain the synchronization of the next CRC calculation results between the master and slave devices, the CRC values at both ends should be cleared. It is recommended to clear the CRC value according to the following steps:

1. Disable the SPI module (SPIEN=0);
2. Clear the CRCEN bit (CRCEN=0);
3. Set the CRCEN bit (CRCEN=1);
4. Enable the SPI module (SPIEN=1).

### 27.3.10. SPI interrupts

#### Status flags

- **Transmit packet space available flag (TP)**

This bit is set when the TxFIFO have enough available position to accommodate a packet,

the software can write the next data packet to the TxFIFO by writing the SPI_TDATA register. This bit is clearred when the TxFIFO don't have enough space to place in the next packet, the software can not write the next data packet to the TxFIFO by writing the SPI_TDATA register.

- **Receive packet space available flag (RP)**

This bit is set when the RxFIFO is not empty, which means that at leaset one data packet is received and stored in the receive buffer, and software can read the data packet by reading the SPI_RDATA register. This bit is clearred when the RxFIFO is empty or the data stored in the RxFIFO can not reach the FIFOLVL. So software can not read the data packet by reading the SPI_RDATA register when RxFIFO is empty. Or in this case, the number of remaining RX data frames in the RxFIFO will be indicated by RWNE and RPLVL in the SPI_STAT register, the the application can still read standard number of previous complete packets from RxFIFO without adverse effects.

- **End of transmission/reception flag (ET)**

ET is a status flag to indicate whether the transmission/reception is ongoing or end. After complete transmission, i.e., when TXSIZE data volume is sent or received based on SPI, this flag is set by hardware and can be cleared by software set ETC bit in SPI_STATC register. The ET flag triggers an interrupt at ESTCIE is set to 1.

- **Duplex packet flag (DP)**

If the TP and RP flags are set to 1, the DP flag is set to 1, which means TxFIFO has space for write operations and RxFIFO contains at least one packet for read operations. DP is useful for full-duplex communication, optimizing data upload/download performance, thus minimizing the need for CPU bandwidth and system power, especially when SPI is operating in stop mode.

- **TxFIFO transmission has been filled flag (TXF)**

When all packets of one transsmisson are sent by the application or DMA, which means the TXSIZE data volume has been pushed into TxFIFO, TXF flag will be set to 1 by hardware. This bit can be cleared by the software writes 1 to the TXFC bit of SPI2S_TCRC register. The TXF flag triggers an interrupt at TXFIE is set to 1.

- **Additional number of SPI data to be transacted was reload flag (TXSERF)**

After processing the number of data in TXSIZE, if TXSER contains non-zero value, the contents of TXSER are copied to TXSIZE and the TXSER value is automatically cleared. The transmission will then increase the amount of data corresponding to the newly loaded value in TXSIZE. After the the amount of data have been sent to TxFIFO, the TXSERF flag is set to 1 and triggers an interrupt at TXSERFIE is set to 1.

- **Suspend flag (SPD)**

In master mode, the device automatically suspends the receive mode when the current frame is completed or the RxFIFO is full (MASP is set to 1 in SPI2S_CTL0 register), after MASPR

is executed, SDP is set to 1 by hardware. The SPD flag is set to 1 and triggers an interrupt at ESTCIE is set to 1. The SPD flag can be cleared by writing 1 to SPDC bit in SPI_STATC register.

- **Transmission complete flag (TC)**

This flag is changed by hardware. If TXSIZE = 0, TxFIFO is empty, TC is set to 1, and there is no activity on the bus. If TXSIZE > 0, TC will be set to 1 at the end of the transmission regardless of TxFIFO usage. When TC is set to 1, the transmission is finish. The CRC mode is enabled, TC will be set to 1 after CRC is sent. The TC flag triggers an interrupt at ESTCIE is set to 1.

### Error conditions

- **Configuration error (CONFERR)**

CONFERR is an error flag in master mode. In NSS hardware mode and the NSSDRV is not enabled, the CONFERR is set when the NSS pin is pulled low. In NSS software mode, the CONFERR is set when the NSSI bit is 0. When the CONFERR is set, the SPIEN bit and the MSTMOD bit are cleared by hardware, the SPI is disabled and the device is forced into slave mode. The CONFERR can be cleared by writing 1 to the CONFERRC bit of SPI_STATC register. The CONFERR flag triggers an interrupt at CONFEIE is set to 1.

The SPIEN and MSTMOD bit are write protection until the CONFERR is cleared. The CONFERR bit of the slave cannot be set. In a multi-master configuration, the device can be in slave mode with CONFERR bit set, which means there might have been a multi-master conflict for system control.

- **Receive overrun error (RXORERR)**

The RXORERR bit is set if a data is received when the RxFIFO has not enough space to store this received data. The RxFIFO contents won't be covered with the newly incoming data, so the newly incoming data is lost. The RXORERR flag triggers an interrupt at RXOREIE is set to 1. The RXORERR can be cleared by writing 1 to the RXORERRC bit of SPI_STATC register.

- **Format error (FERR)**

In slave TI mode, the slave also monitors the NSS signal and set an error flag if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte. The FERR flag triggers an interrupt at FEIE is set to 1. The FERR can be cleared by writing 1 to the FERRC bit of SPI_STATC register.

- **CRC error (CRCERR)**

When the CRCEN bit is set, the CRC calculation result of the received data in the SPI_RCRC register is compared with the received CRC value after the last data, the CRCERR is set when they are different. The CRCERR flag triggers an interrupt at CRCERIE is set to 1. The CRCERR can be cleared by writing 1 to the CRCERRC bit of SPI_STATC register.

■ **Transmit underrun error (TXURERR)**

In slave transfer mode, if the TxFIFO is empty, and need to send new data into the transfer shift register, the TXURERR bit is set. After an underrun is caught, the next data supplied for sending depends on the TXUROP, WORDEN, BYTEN bit. The TXURERR flag triggers an interrupt at TXUREIE is set to 1. The TXURERR can be cleared by writing 1 to the TXURERRC bit of SPI_STATC register.

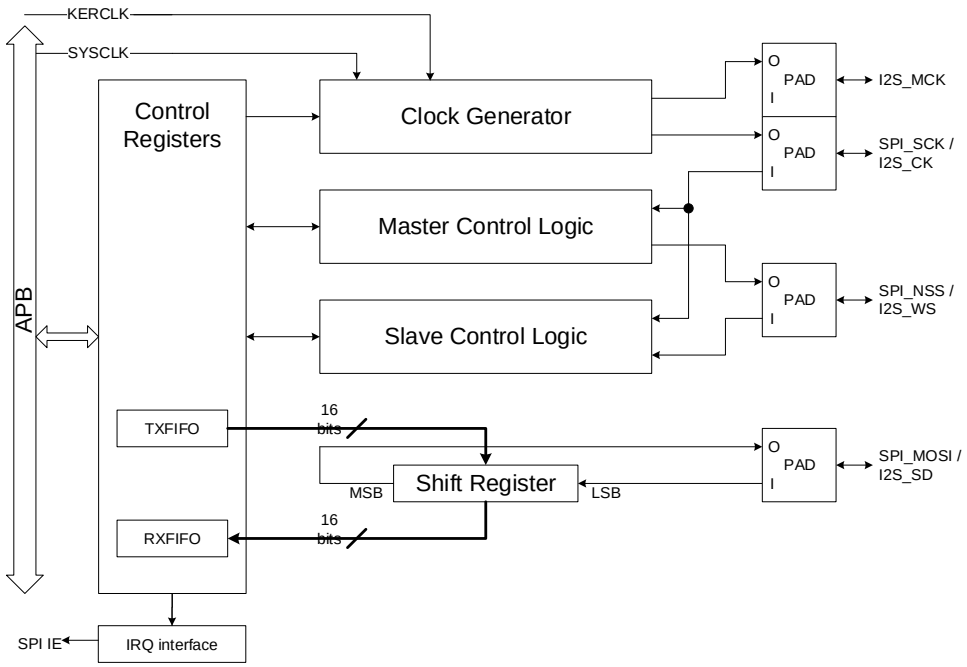
**Table 27-8. SPI interrupt requests**

| Flag | Description | Clear method | Interrupt enable bit |
|---|---|---|---|
| TP | Transmit packet space available flag | TP is cleared by hardware when TxFIFO space less than FIFOLVL | TPIE |
| RP | Receive packet space available flag | RP is cleared by hardware when the data in RxFIFO less than FIFOLVL | RPIE |
| ET | End of transmission/reception flag | ETC set to 1 | ESTCIE |
| DP | Duplex packet flag | DP is cleared by hardware when TP and RP are cleared | DPIE |
| TXF | TxFIFO transmission has been filled flag | TXFC set to 1 | TXFIE |
| TXSERF | Additional number of SPI data to be transacted was reload flag | TXSERFC set to 1 | TXSERFIE |
| SPD | Suspend flag | SPDC set to 1 | ESTCIE |
| TC | Transmission complete flag | TC is cleared by hardware when transmission start | ESTCIE |
| CONFERR | Configuration error | CONFERRC set to 1 | CONFEIE |
| RXORERR | Receive overrun error | RXORERRC set to 1 | RXOREIE |
| FERR | Format error | FERRC set to 1 | FEIE |
| CRCERR | CRC error | CRCERRC set to 1 | CRCERIE |
| TXURERR | Transmit underrun error | TXURERRC set to 1 | TXUREIE |

# 27.4. I2S function overview

## 27.4.1. I2S block diagram

**Figure 27-19. Block diagram of I2S**




- SYSCLK: system clock, provided by APB bus.
- KERCLK: kernel clock, provided by RCU. It has an asynchronous relationship with the system clock.
- The clock signal frequency must be consistent with user conditions and data transmission speed to prevent data loss. (**NOTE**: suggest the frequency of SYSCLK is greater than or equal to KERCLK)
- The SCK signal of the I2S slave is provided by the I2S master.

There are five sub modules to support I2S function, including control registers, clock generator, master control logic, slave control logic and shift register. All the user configuration registers are implemented in the control registers module, including the TxFIFO and RxFIFO. The clock generator is used to produce I2S communication clock in master mode. This clock generator is the source of MCK. The master control logic is implemented to generate the I2S_WS signal and control the communication in master mode. The slave control logic is implemented to control the communication in slave mode according to the received I2SCK and I2S_WS. The shift register handles the serial data transmission and reception on I2S_SD.

## 27.4.2. I2S signal description

There are four pins on the I2S interface, including I2S_CK, I2S_WS, I2S_SD and I2S_MCK. I2S_CK is the serial clock signal, which shares the same pin with SPI_SCK. I2S_WS is the frame control signal, which shares the same pin with SPI_NSS. I2S_SD is the serial data signal, which shares the same pin with SPI_MOSI. I2S_MCK is the master clock signal. It

produces a frequency rate equal to 256 x Fs, and Fs is the audio sampling frequency.

### 27.4.3. I2S audio standards

The I2S audio standard is selected by the I2SSTD bits in the SPI_I2SCTL register. Four audio standards are supported, including I2S Philips standard, MSB justified standard, LSB justified standard, and PCM standard. All standards except PCM handle audio data time-multiplexed on two channels (the left channel and the right channel). For these standards, the I2S_WS signal indicates the channel side. For PCM standard, the I2S_WS signal indicates frame synchronization information.

The data length and the channel length are configured by the DTLEN bits and CHLEN bit in the SPI_I2SCTL register. Since the channel length must be greater than or equal to the data length, four packet types are available. They are 16-bit data packed in 16-bit frame, 16-bit data packed in 32-bit frame, 24-bit data packed in 32-bit frame, and 32-bit data packed in 32-bit frame.

For all standards and packet types, the most significant bit (MSB) is always sent first. For all standards based on two channels time-multiplexed, the channel left is always sent first followed by the channel right.

#### I2S Philips standard

For I2S Philips standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK, and I2S_WS becomes valid one clock before the data. The timing diagrams for each configuration are shown below.

**Figure 27-20. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0)**

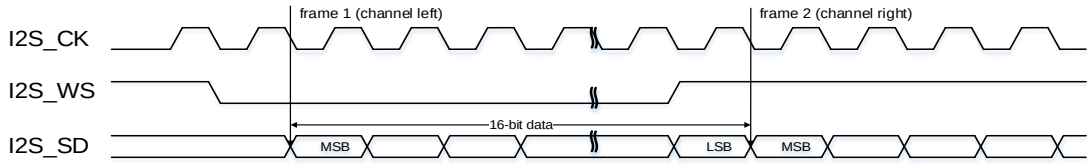



**Figure 27-21. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1)**

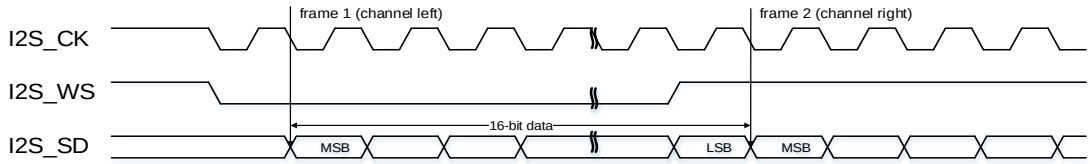



**Figure 27-22. I2S Philips standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0)**

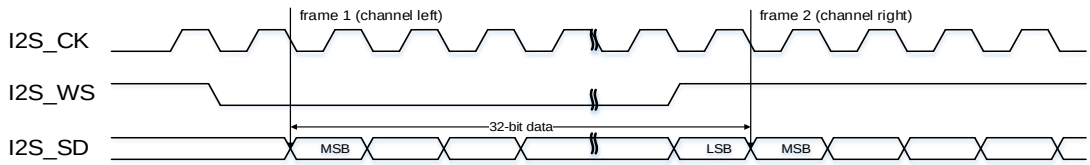


**Figure 27-23. I2S Philips standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1)**

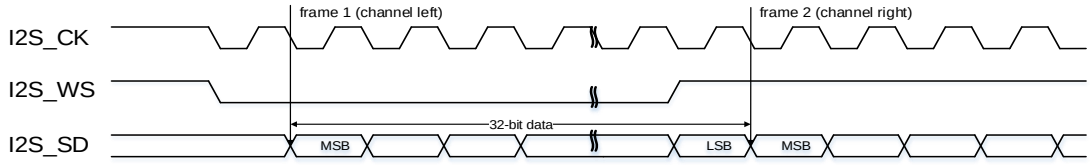



**Figure 27-24. I2S Philips standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)**

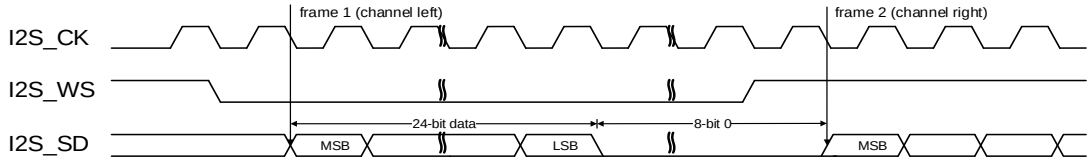



**Figure 27-25. I2S Philips standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)**

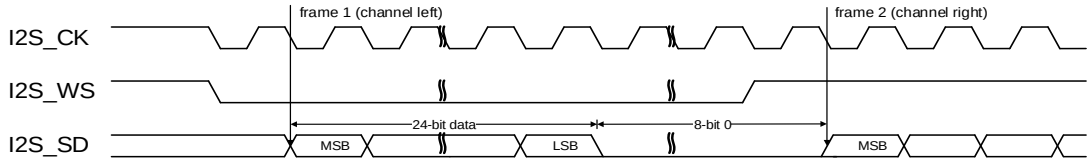



**Figure 27-26. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)**

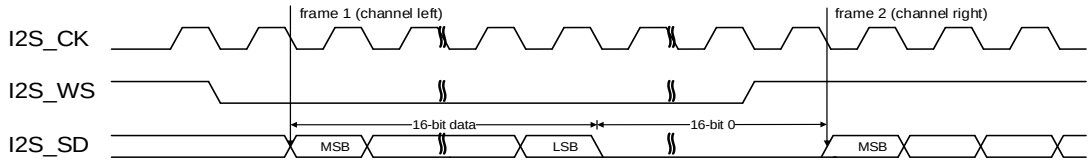
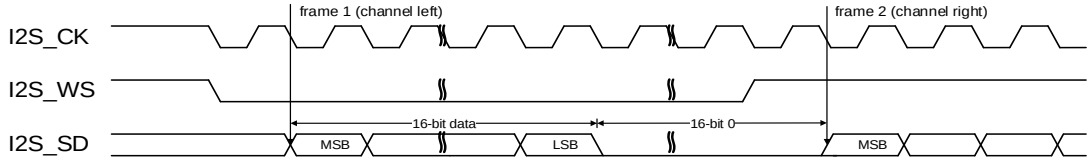



**Figure 27-27. I2S Philips standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)**




## MSB justified standard

For MSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. The timing diagrams for each configuration are shown below.

**Figure 27-28. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0)**

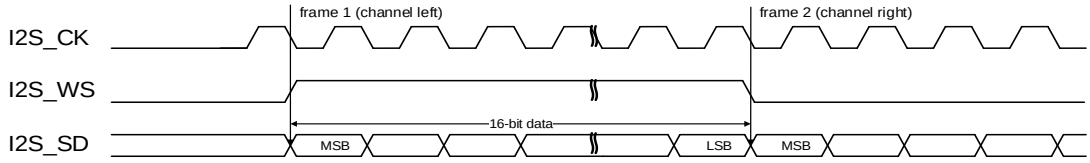


**Figure 27-29. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1)**

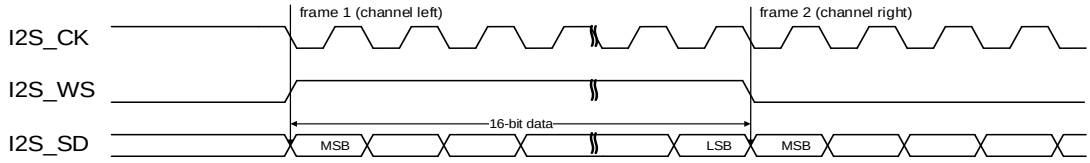



**Figure 27-30. MSB justified standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0)**

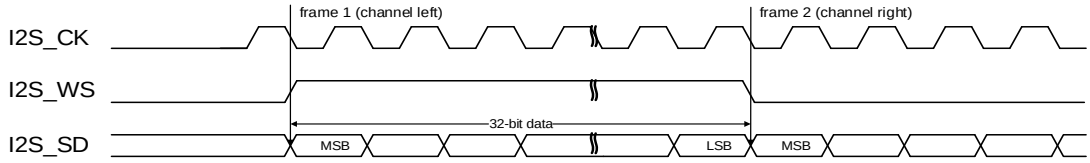



**Figure 27-31. MSB justified standard timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1)**

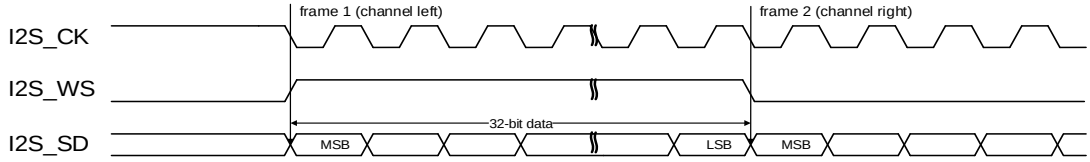



**Figure 27-32. MSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)**

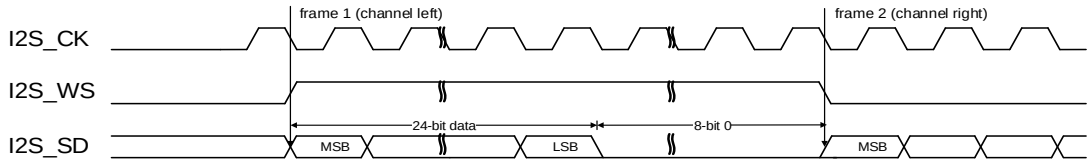



**Figure 27-33. MSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)**

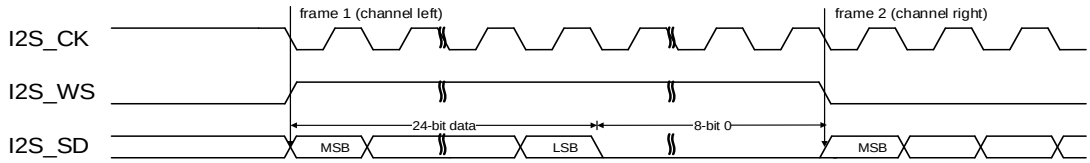



**Figure 27-34. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)**

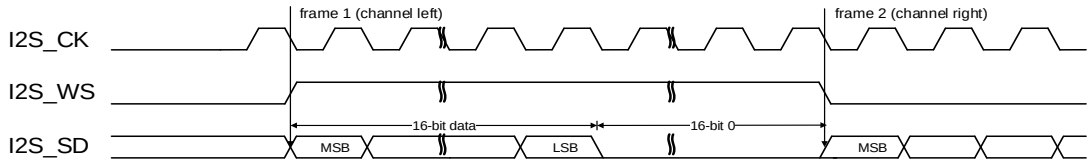
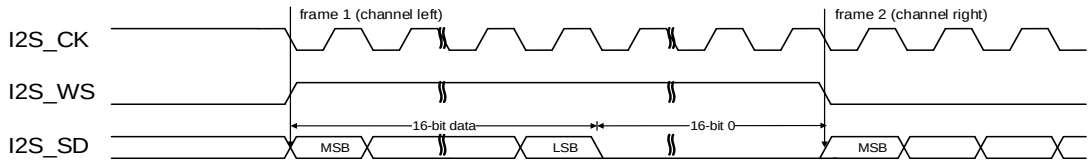


**Figure 27-35. MSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)**




## LSB justified standard

For LSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. In the case that the channel length is equal to the data length, LSB justified standard and MSB justified standard are exactly the same. In the case that the channel length is greater than the data length, the valid data is aligned to LSB for LSB justified standard while the valid data is aligned to MSB for MSB justified standard. The timing diagrams for the cases that the channel length is greater than the data length are shown below.

**Figure 27-36. LSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)**

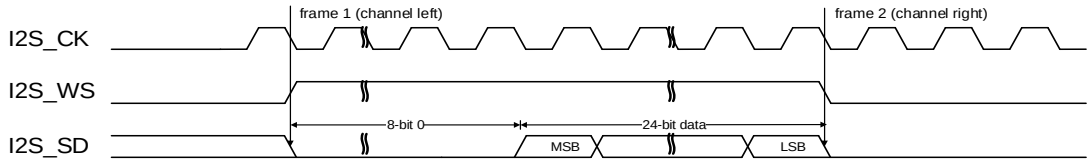



**Figure 27-37. LSB justified standard timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)**

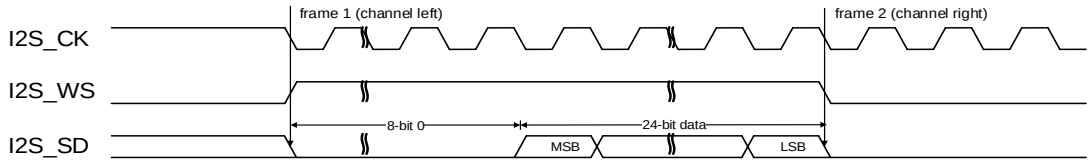



**Figure 27-38. LSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)**

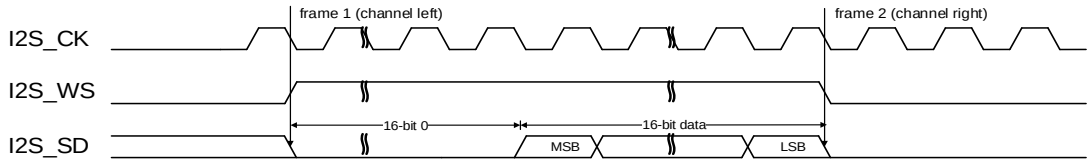
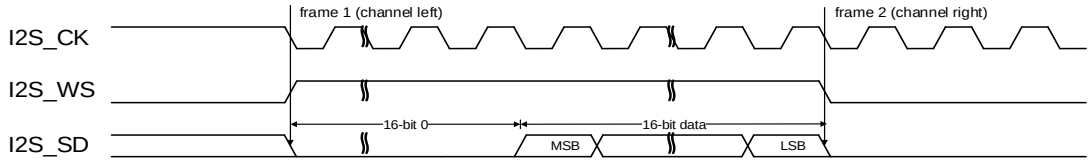



**Figure 27-39. LSB justified standard timing diagram (DTLEN = 00, CHLEN = 1, CKPL =**

1)




## PCM standard

For PCM standard, I2S_WS and I2S_SD are updated on the rising edge of I2S_CK, and the I2S_WS signal indicates frame synchronization information. Both the short frame synchronization mode and the long frame synchronization mode are available and configurable using the PCMSMOD bit in the SPI_I2SCTL register. The SPI_TDATA / SPI_RDATA register is handled in the exactly same way as that for I2S Philips standard. The timing diagrams for each configuration of the short frame synchronization mode are shown below.

**Figure 27-40. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 0)**

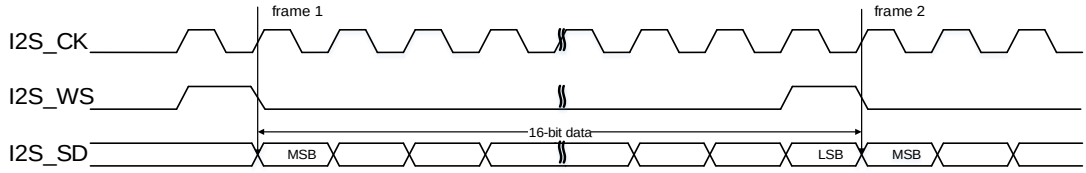



**Figure 27-41. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1)**

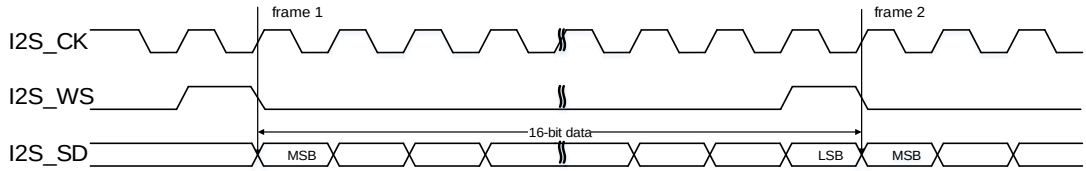



**Figure 27-42. PCM standard short frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0)**

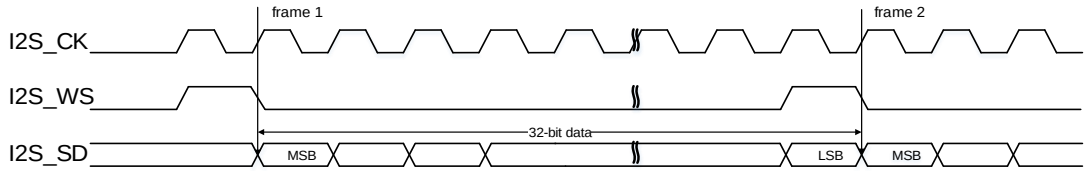



**Figure 27-43. PCM standard short frame synchronization mode timing diagram (DTLEN**

**= 10, CHLEN = 1, CKPL = 1)**

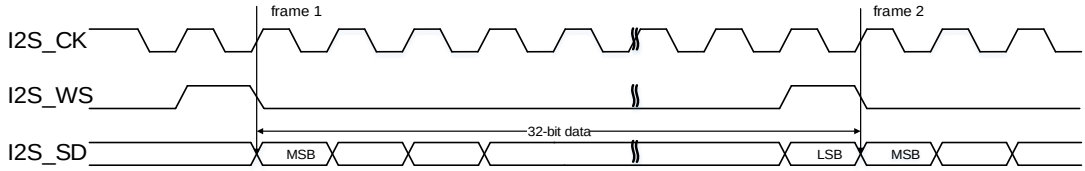



**Figure 27-44. PCM standard short frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)**

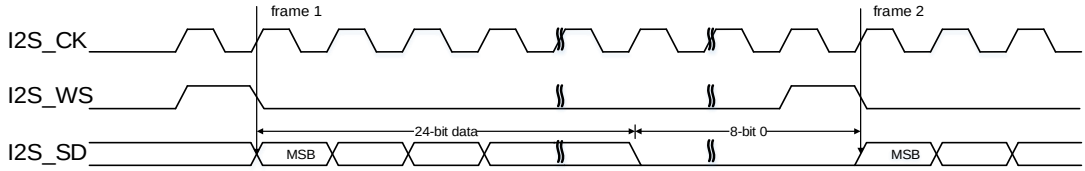



**Figure 27-45. PCM standard short frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)**

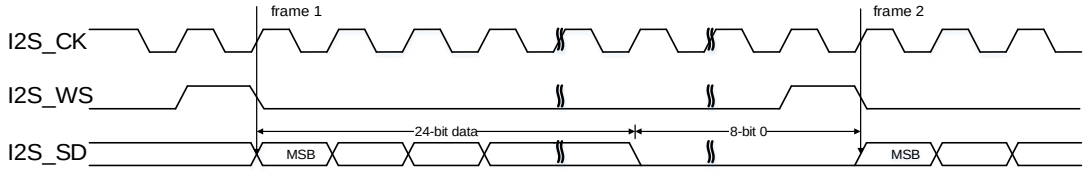



**Figure 27-46. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)**

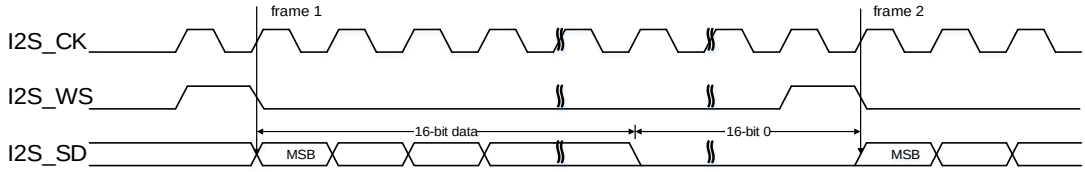
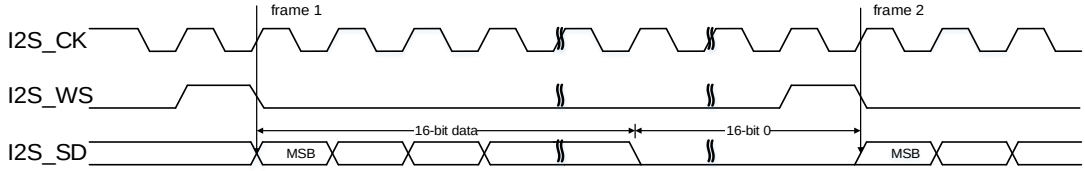



**Figure 27-47. PCM standard short frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)**




The timing diagrams for each configuration of the long frame synchronization mode are shown below.

**Figure 27-48. PCM standard long frame synchronization mode timing diagram (DTLEN**

**= 00, CHLEN = 0, CKPL = 0)**

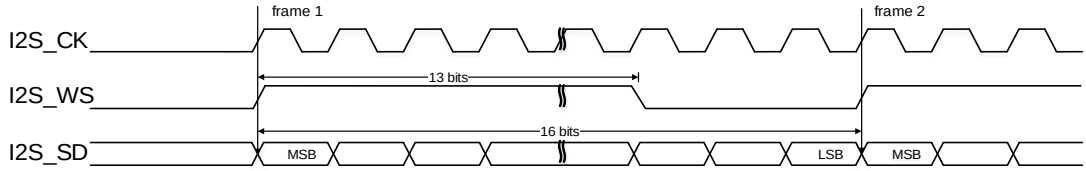



**Figure 27-49. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 0, CKPL = 1)**

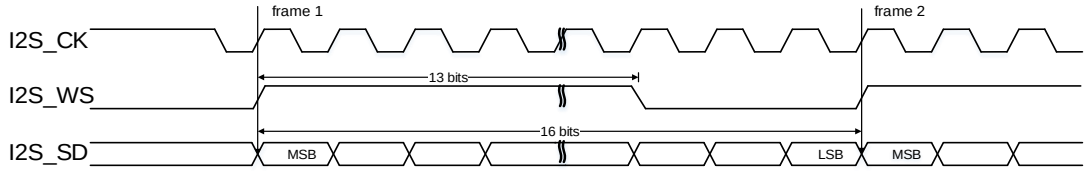



**Figure 27-50. PCM standard long frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 0)**

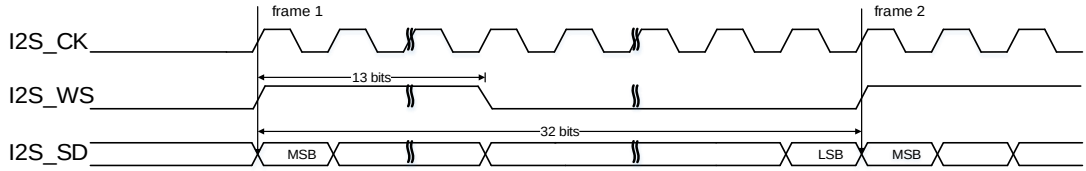



**Figure 27-51. PCM standard long frame synchronization mode timing diagram (DTLEN = 10, CHLEN = 1, CKPL = 1)**

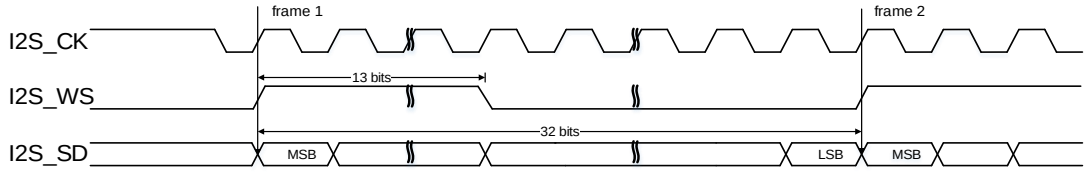



**Figure 27-52. PCM standard long frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 0)**

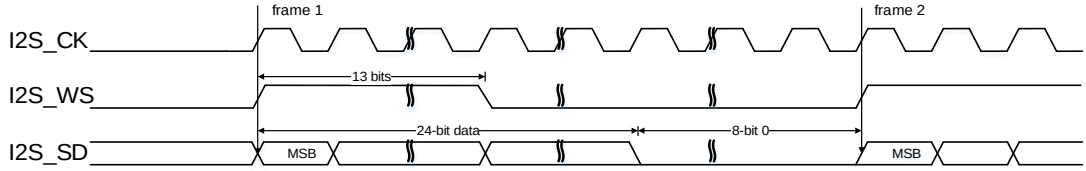



**Figure 27-53. PCM standard long frame synchronization mode timing diagram (DTLEN = 01, CHLEN = 1, CKPL = 1)**

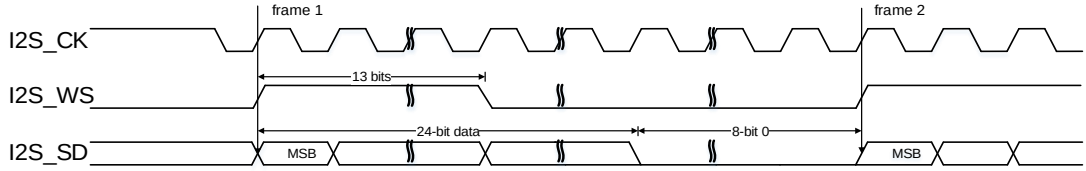


**Figure 27-54. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 0)**

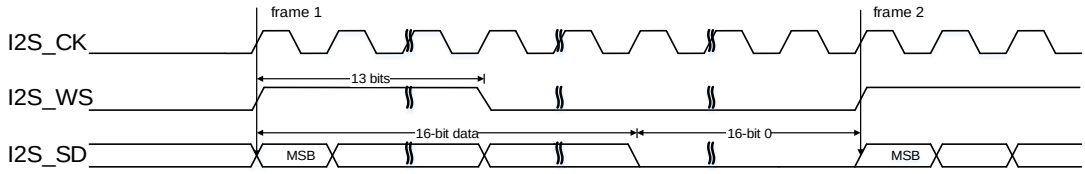
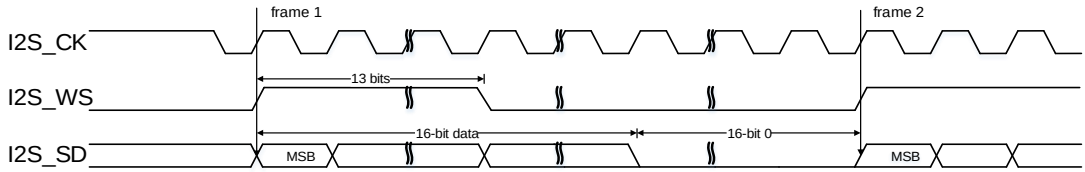
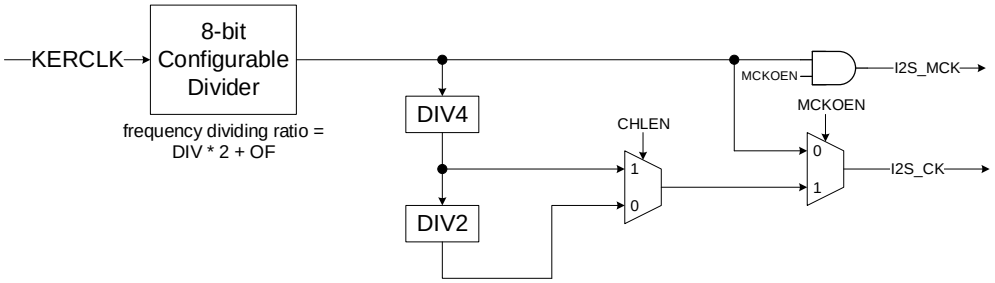



**Figure 27-55. PCM standard long frame synchronization mode timing diagram (DTLEN = 00, CHLEN = 1, CKPL = 1)**




## 27.4.4. I2S clock

**Figure 27-56. Block diagram of I2S clock generator**




The block diagram of I2S clock generator is shown as ***Figure 27-56. Block diagram of I2S clock generator***. The I2S interface clocks are configured by the DIV bits, the OF bit, the MCKOEN bit and the CHLEN bit in the SPI_I2SCTL register. The source clock is the kernel clock (KERCLK). The I2S bitrate can be calculated by the formulas shown in ***Table 27-9. I2S bitrate calculation formulas***.

**Table 27-9. I2S bitrate calculation formulas**

| MCKOEN | CHLEN | Formula |
|---|---|---|
| 0 | 0 | KERCLK / (DIV * 2 + OF) |
| 0 | 1 | KERCLK / (DIV * 2 + OF) |
| 1 | 0 | KERCLK / (8 * (DIV * 2 + OF)) |
| 1 | 1 | KERCLK / (4 * (DIV * 2 + OF)) |

The relationship between audio sampling frequency (Fs) and I2S bitrate is defined by the following formula:

Fs = I2S bitrate / (number of bits per channel * number of channels)

So, in order to get the desired audio sampling frequency, the clock generator needs to be configured according to the formulas listed in ***Table 27-10. Audio sampling frequency***

*calculation formulas*.

**Note:** It is recommended that the configuration value of the I2S serial clock be set to less than 1/6 of the PCLK clock corresponding to the APB to which the I2S is mounted (excluding 1/6).

**Table 27-10. Audio sampling frequency calculation formulas**

| MCKOEN | CHLEN | Formula |
|---|---|---|
| 0 | 0 | KERCLK / (32 * (DIV * 2 + OF)) |
| 0 | 1 | KERCLK / (64 * (DIV * 2 + OF)) |
| 1 | 0 | KERCLK / (256 * (DIV * 2 + OF)) |
| 1 | 1 | KERCLK / (256 * (DIV * 2 + OF)) |

### 27.4.5. RxFIFO and TxFIFO

RxFIFO and TxFIFO are used in different directions for I2S data transactions, and they can enable the I2S to work in a continuous flow, and can prevent short data frame length or interrupt/DMA overrun occurs when the delay is too long.

A write access to the SPI_TDATA register stores the written data in the end of TxFIFO, while a read access to the SPI_RDATA returns the oldest value in RxFIFO which has not been read. In I2S mode, left audio sampling and right audio sampling are interleaved in the FIFO. This means that for the send operation, the user must start filling TxFIFO with left samples and then right samples, and so on. For receive mode, the first data read from RxFIFO is left channel, the next data is the right channel, and so on.

FIFO processing depends on the data length (DTLEN value), the size of the access to the FIFO register (8, 16, or 32 bits). The size of TxFIFO / RxFIFO is 16 x 32 bits and the maximum access data frame length is 32 bits. According to different frame length, the maximum number of frames that can be stored in FIFO is described in *Table 27-11. The maxinum number of data frame stored in I2SX FIFO*. (N = FIFO size / 32 = 16 x 32 / 32 = 16)

**Table 27-11. The maxinum number of data frame stored in I2SX FIFO**

| Data length (DTLEN) | DTLEN = 16 bit | DTLEN = 24 bit | DTLEN = 32 bit |
|---|---|---|---|
| Frame numbers （WORDEN = 0） | N | - | - |
| Frame numbers （WORDEN = 1） | 2N | N | N |

According to the FIFO threshold of programmable can generate interrupts or DMA requests.The influence of FIFOLVL is consistent with SPI.

**NOTE**: SPI_TDATA and SPI_RDATA data is the default right aligned. Both RxFIFO and TxFIFO content is kept flushed when I2S is disable (I2SEN = 0).

## 27.4.6. Operation

### Operation modes

The operation mode is selected by the I2SOPMOD bits in the SPI_I2SCTL register. There are four available operation modes, including master transmission mode, master reception mode, slave transmission mode, and slave reception mode. The direction of I2S interface signals for each operation mode is shown in the ***Table 27-12. Direction of I2S interface signals for each operation mode***.

**Table 27-12. Direction of I2S interface signals for each operation mode**

| Operation mode | I2S_MCK | I2S_CK | I2S_WS | I2S_SD |
|---|---|---|---|---|
| Master transmission | Output or NU(1) | Output | Output | Output |
| Master reception | Output or NU(1) | Output | Output | Input |
| Slave transmission | Input or NU(1) | Input | Input | Output |
| Slave reception | Input or NU(1) | Input | Input | Input |

(1) NU means the pin is not used by I2S and can be used by other functions.

### I2S initialization sequence

I2S initialization sequence contains five steps shown below. In order to initialize I2S working in master mode, all the five steps should be done. In order to initialize I2S to slave mode, only step 2, step 3, step 4, step 5 and step 6 should be done.

- Step 1: Configure the DIV [7:0] bits, the OF bit, and the MCKOEN bit in the SPI_I2SCTL register, in order to define the I2S bitrate and determine whether I2S_MCK needs to be provided or not.
- Step 2: Configure the CKPL in the SPI_I2SCTL register, in order to define the idle state clock polarity.
- Step 3: Configure the FIFO level (FIFOLVL[3:0] bits in the SPI_CFG0 register).
- Step 4: Configure the I2SSEL bit, the I2SSTD [1:0] bits, the PCMSMOD bit, the I2SOPMOD [1:0] bits, the DTLEN [1:0] bits, and the CHLEN bit in the SPI_I2SCTL register to define the I2S feature.
- Step 5: Configure the TPIE bit, the RPIE bit, the TXUREIE bit, the RXOREIE bit, the FEIE bit, the DMATEN bit, and the DMAREN bit, in order to select the potential interrupt sources and the DMA capabilities. This step is optional.
- Step 6: Set the I2SEN bit in the SPI_I2SCTL register to enable I2S.
- Step 7: Activate the serial interface by setting MSTART in the SPI_CTL0 register to 1.

### I2S basic transmission and reception sequence

#### Transmission sequence

After the initialization sequence, the I2S is enabled and stays at idle state. In master mode, the transmission starts when the application writes a data into the TxFIFO. The application should ensure that data is already written into TxFIFO before the transmission starts in slave

mode.

When I2S begins to send a data frame, it first loads this data frame from the TxFIFO to the shift register and then begins to transmit the loaded data frame.

Write access to SPI_TDATA is managed by the TP event. With the TP flag set to 1, the application performs an appropriate number of I2S data register writes to transfer the contents of the data package. After uploading the new complete package, the application checks the TP value to check whether TxFIFO can receive additional data packet, if TP = 1, they are uploaded packet by packet until TP reads 0. If the transmission size and packet size are not aligned, when the last number of data packets to be transferred cannot reach the configured size (set by FIFOLVL). The application can still write standard number of previous complete packets to TxFIFO without adverse effects: only consistent data (complete data frames) will pull down to TxFIFO, while redundant write times (or any incomplete data) will be ignored.

In master mode, software should write the next data into SPI_TDATA register before the transmission of current data frame is completed if it desires to generate continuous transmission. As long as there is data in TxFIFO, data delivery continues until TxFIFO becomes empty.

**Reception sequence**

After the last valid sample clock, the incoming data will be moved from shift register to the RxFIFO and RP will be set to 1. The application should read SPI_RDATA register to get the received data and this will clear the RP flag automatically when the number of data less than FIFOLVL in RxFIFO.

A read access to SPI_RDATA is managed by the RP event. With the RP flag set to 1, the application performs an appropriate number of SPI data register reads to download a single piece of data contents of the package. After the complete packet is downloaded, the application checks the RP value to see whether there are other packets in the RxFIFO, if any, they are downloaded packet by packet until RP reads 0. At the end of the receive, it may occur that some data is still available in the RxFIFO without reaching the FIFOLVL level, so RP will not be set to 1. In this case, the number of remaining RX data frames in the RxFIFO will be indicated by RWNE and RPLVL in the SPI_STAT register. If the transmission size and packet size are not aligned, the above condition occurs when the last number of data packets to be received cannot reach the configured size (set by FIFOLVL). However, the application can still read standard number of previous complete packets from RxFIFO without adverse effects: only consistent data (complete data frames) will pull up from RxFIFO, while redundant read times (or any incomplete data) will read 0.

### I2S disabling sequence

I2S master disabling sequence:

- Step 1: Stop the bus clock and DMA function.
- Step 2: Clear the I2SEN bit in the SPI_CTL0 to stop the I2S peripheral.

I2S slave disabling sequence:

- Step 1: Clear the I2SEN bit in the SPI_CTL0 to stop the I2S peripheral.
- Step 2: Stop the bus clock and DMA function.

### 27.4.7. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

### 27.4.8. I2S interrupts

#### Status flags

There are two status flags implemented in the SPI_STAT register (TP, RP), There is one status flag implemented in the SPI_I2SCTL register (I2SCH). The user can use them to fully monitor the state of the I2S bus.

- **Transmit packet space available flag (TP)**

This bit is set when the TxFIFO have enough available position to accommodate a packet, the software can write the next data packet to the TxFIFO by writing the SPI_TDATA register. This bit is clearred when the TxFIFO don't have enough space to place in the next packet, the software can not write the next data packet to the TxFIFO by writing the SPI_TDATA register.

- **Receive packet space available flag (RP)**

This bit is set when the RxFIFO is not empty, which means that at leaset one data packet is received and stored in the receive buffer, and software can read the data packet by reading the SPI_RDATA register. This bit is clearred when the RxFIFO is empty or the data stored in the RxFIFO can not reach the FIFOLVL. So software can not read the data packet by reading the SPI_RDATA register when RxFIFO is empty. Or in this case, the number of remaining RX data frames in the RxFIFO will be indicated by RWNE and RPLVL in the SPI_STAT register, the the application can still read standard number of previous complete packets from RxFIFO without adverse effects can still read standard number of previous complete packets from RxFIFO without adverse effects.

- **I2S channel flag (I2SCH)**

This flag indicates the channel side information of the current transfer and has no meaning in PCM mode. It is updated when channel switch in transmission mode or in reception mode. This flag doesn’t generate any interrupt.

**Note**: because of the existence of FIFO, the change of this bit have no relevant with TP / RP flag, and the change will happen at the end of channel transmission (channel transmission complete, which doesn't mean data transmission complete. For example, channel with 32 bits, data with 16 bits, channel transmission complete on behalf of the 32-bit data completed).

### Error conditions

There are three error flags:

- **Receive overrun error (RXORERR)**

The RXORERR bit is set if a data is received when the RxFIFO has not enough space to store this received data. The RxFIFO contents won't be covered with the newly incoming data, so the newly incoming data is lost. The RXORERR flag triggers an interrupt at RXOREIE is set to 1. The RXORERR can be cleared by writing 1 to the RXORERRC bit of SPI_STATC register.

Note: I2S mode has a hardware mechanism to prevent the error of data interchange between left and right channels due to overrun. For example, data receive sequence is L0->R0->L1->R1->L2->R2->L3->R3... LN->RN (L for left-channel data, R for right-channel data). When the overrun occurs after R1 is received, L2 data is lost. When RxFIFO recovers (which can receive data), the hardware will automatically discard R2 data, and receive L3 data to the left channel, and receive R3 data to the right channel. When the overrun occurs after L2 is received, R2 data is lost. When RxFIFO recovers (which can receive data), the hardware will automatically discard L3 data and receive R3 data to the right channel and receive L4 data to the left channel.

- **Format error (FERR)**

In slave I2S mode, the slave also monitors the I2S_WS signal and set an error flag if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte. The FERR flag triggers an interrupt at FEIE is set to 1. The FERR can be cleared by writing 1 to the FERRC bit of SPI_STATC register.

- **Transmit underrun error (TXURERR)**

In slave transfer mode, if the TxFIFO is empty, and need to send new data into the transfer shift register, the TXURERR bit is set. When this happens, at least one data is lost. The TXURERR flag triggers an interrupt at TXUREIE is set to 1. The TXURERR can be cleared by writing 1 to the TXURERRC bit of SPI_STATC register.

Note: I2S mode has a hardware mechanism to prevent the left and right channel data interchange error due to underrun. For example, data transfer sequence is L0->R0->L1->R1->L2->R2->L3->R3... LN->RN (L for left-channel data, R for right-channel data). Current underrun occurs after R1 is sent, L2 data is not timely transferred into TxFIFO resulting in TxFIFO is empty, the hardware will automatically transfer R1 data into the left channel, and then transfer R1 data to the right channel. When L2 data is transferred into TxFIFO, and then transfer L2 data to the left channel, and transfer R2 data to the right channel. When the underrun occurs after L2 is sent, R2 data is not timely transferred into TxFIFO resulting in TxFIFO is empty, the hardware will automatically transfer L2 data to the right channel, and then transfer L2 data to the left channel, when R2 data is transferred into TxFIFO, and then transfer R2 data to the right channel, and then transfer L3 data to the left channel.

I2S interrupt events and corresponding enabled bits are summed up in the ***Table 27-13. I2S***

*interrupt*.

**Table 27-13. I2S interrupt**

| Interrupt flag | Description | Clear method | Interrupt enable bit |
|---|---|---|---|
| TP | Transmit packet space available flag | TP is cleared by hardware when TxFIFO contains less than FIFOLVL empty locations | TPIE |
| RP | Receive packet space available flag | RP is cleared by hardware when RxFIFO contains less than FIFOLVL empty locations | RPIE |
| TXURERR | Transmit underrun error | TXURERRC set to 1 | TXUREIE |
| RXORERR | Receive overrun error | RXORERRC set to 1 | RXOREIE |
| FERR | Format error | FERRC set to 1 | FEIE |

## 27.5. Register definition

SPI0/I2S0 base address: 0x4001 3000

SPI1/I2S1 base address: 0x4000 3800

SPI2/I2S2 base address: 0x4000 3C00

SPI3 base address: 0x4001 3400

SPI4 base address: 0x4001 5000

SPI5/I2S5 base address: 0x4001 3800

### 27.5.1. Control register 0 (SPI_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | IOAFEN |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXCRCI | RXCRCI | CRCFS | NSSI | Reserved | MSPDR | MSTART | MASP | Reserved | | | | | | | SPIEN |
| rw | rw | rw | rw | | w | rw | rw | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | IOAFEN | Related IOs AF configuration enable<br>0: Related IOs AF configuration is enable<br>1: Related IOs AF configuration is disable<br>This bit is set by software and cleared by hardware whenever SPIEN bit is changed from 1 to 0, or it can be cleared to 0 by software when the SPIEN bit and the CONFERR bit are 0. It is cleared and cannot be set when the CONFERR bit is set. This bit will be write protected when SPIEN is set. The SPI_CFG1 register cannot be changed when this bit is set. |
| 15 | TXCRCI | The transmitter CRC initialization configuration<br>0: All 0 mode<br>1: All 1 mode |
| 14 | RXCRCI | The receiver CRC initialization configuration<br>0: All 0 mode<br>1: All 1 mode |

| | | |
|---|---|---|
| 13 | CRCFS | Full scale CRC polynomial configuration<br>0: Not use full scale CRC polynomial<br>1: Use full scale CRC polynomial |
| 12 | NSSI | The input level of internal NSS signal<br>0: NSS pin is pulled low<br>1: NSS pin is pulled high<br>Only when the NSSIM position 1, this bit is valid. This bit values will effect to the peripheral NSS input, and ignore the I/O values of NSS pin. |
| 11 | Reserved | Must be kept at reset value. |
| 10 | MSPDR | Suspend request in SPI master mode<br>0: Do not request suspend<br>1: Request suspend<br>To read this bit is zero. In SPI master mode, if this bit is set by software, SPI communication will be suspended and the MSTART bit will reset when the current frame transfer over. Application should check SPD flag in SPI_STAT register to check the end of transaction. Before SPI disabled, master communication must be suspended (either by setting this bit or emptying the SPI_TDATA register). |
| 9 | MSTART | Master start transfer<br>0: The master transfer is in idle status<br>1: The master transfer is occurring, or has been temporarily suspended by automatic suspend<br>This bit can be set by software, and can be clearred by hardware when ET flag in SPI_STAT register equal to 1 or when receiving suspend request.<br>In SPI mode, only when SPIEN = 1 and MSTMOD = 1 (in SPI_CFG1 register), this bit can be set.<br>In I2S / PCM mode, only when I2SEN = 1, this bit can be set. |
| 8 | MASP | The master is suspended automatically in receive mode<br>0: SPI stream/clock generation is continuous whether or not an overrun occurs<br>1: Before the overrun condition is reached, the SPI stream is suspended in the full RxFIFO state. SPD flag (in SPI_STAT register) will set 1<br>When SPI communication is suspended to prevent an overrun condition, several bits of the next frame may be synchronized out due to an internal synchronization delay. After reading the RxFIFO, communication resumes and subsequent bits transfers continue without any restrictions.<br>For the same reason, automatic suspension is not very reliable when the data size is less than 8 bits. In this case, by combining the insert delay between data frames applied when MDFD parameters remain non-zero value to achieve safe suspension; The sum of data size and interleaving SPI cycles should always result in an interval of at least 8 SPI clock cycles.<br>**NOTE:** MASP can be set only in receiving mode, otherwise it may cause RXORERR (reception overrun) error. |

| | | |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | SPIEN | SPI enable<br>0: SPI peripheral is disabled<br>1: SPI peripheral is enabled<br>This bit can be set and cleared by software and can not be set when CONFERR bit is set in SPI_STAT register. |

### 27.5.2. Control register 1 (SPI_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXSER[15:0] | | | | | | | | | | | | | | | |
| rs | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TXSIZE[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | TXSER[15:0] | When previous number of data stored in the TXSIZE has been transferred, it will reload the transmission expansion data amount in the TXSER to TXSIZE.<br>These bits can only be set by the software when its value is zero. After the TXSIZE reload, it is cleared by the hardware. If it is the last time TXSIZE reload, the TXSER counter must to be previously written before CTXSIZE (in SPI_STAT register) counter reaches 1, in addition to this last reload, must program the TXSER value in advance before the CTXSIZE (in SPI_STAT register) counter reaches 1 (or reaches 2 if the CRCEN is set) and CTXSIZE counter is less than the last TXSER value minus 1, otherwise reloads will not be considered and communication will be terminated normally.<br>**NOTE:** TXSER value need to be set greater than 1. |
| 15:0 | TXSIZE[15:0] | The current number of data to transfer<br>These bits can be modified by software, and can not be modified when MSTART bit set 1. When TXSIZE is 0, and MSTART is set to 1, it will start endless transmission. When CRC is enable, TXSIZE cannot be set to 0xFFFF / 0x0001. |

### 27.5.3. Configuration register 0 (SPI_CFG0)

Address offset: 0x08
Reset value: 0x0007 0007

This register can be accessed by word (32-bit). When SPI is enable, this register is write

protected, except DMATEN and DMAREN bits.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | PSC[2:0] | | | Reserved | | | WORDEN | BYTEN | CRCEN | Reserved | CRCSZ[4:0] | | | | |
| | rw | | | | | | rw | rw | rw | | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMATEN | DMAREN | Reserved | TXURDT[1:0] | | TXUROP[1:0] | | FIFOLVL[3:0] | | | | DZ[4:0] | | | | |
| rw | rw | | rw | | rw | | rw | | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:28 | PSC[2:0] | Master clock prescaler selection<br>000: KERCLK / 2<br>001: KERCLK / 4<br>010: KERCLK / 8<br>011: KERCLK / 16<br>100: KERCLK / 32<br>101: KERCLK / 64<br>110: KERCLK / 128<br>111: KERCLK / 256<br>**NOTE:** the 000 / 001 configuration of PSC should not be used in Quad-SPI mode.<br>**NOTE:** the 000 configuration of PSC should not be used in TI mode. |
| 27:25 | Reserved | Must be kept at reset value. |
| 24 | WORDEN | Word access mode enable<br>This bit is used to indicate the width of access to the FIFO, and to generate the threshold of the RxFIFO for the RWNE.<br>0: According to BYTEN mode<br>1: Word access mode enable<br>In I2S mode,in order to ensure the stability of channel data, WORDEN is associated with DTLEN.<br>If DTLEN = 0, WORDEN must be 0.<br>If DTLEN > 0, WORDEN must be 1. |
| 23 | BYTEN | Byte access mode enable<br>This bit is used to indicate the width of access to the FIFO, and to generate the threshold of the RxFIFO for the RWNE.<br>0: Halfword access mode enable<br>1: Byte access mode enable<br>in I2S mode, BYTEN must always be 0 to ensure stable channel data.. |
| 22 | CRCEN | CRC calculation enable<br>0: CRC calculation is disabled<br>1: CRC calculation is enabled |
| 21 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 20:16 | CRCSZ[4:0] | CRC size<br>This bit fields must be equal to DZ value or multiples of DZ value.<br>00000: Not use<br>00001: Not use<br>00010: Not use<br>00011: 4-bit<br>00100: 5-bit<br>00101: 6-bit<br>…..<br>11101: 30-bit<br>11110: 31-bit<br>11111: 32-bit |
| 15 | DMATEN | Transmit buffer DMA enable<br>0: Transmit buffer DMA is disabled<br>1: Transmit buffer DMA is enabled |
| 14 | DMAREN | Receive buffer DMA enable<br>0: Receive buffer DMA is disabled<br>1: Receive buffer DMA is enabled |
| 13 | Reserved | Must be kept at reset value. |
| 12:11 | TXURDT[1:0] | Detection of underrun error at slave transmitter<br>00: The underrun detected condition is set at the beginning of the data frame (no first bit protection)<br>01: The underrun detected condition is set at the end of last data frame<br>10: The underrun detected condition is set at the beginning of NSS signal<br>11: Reserved |
| 10:9 | TXUROP[1:0] | Operation of slave transmitter when underrun detected<br>00: Slave send a constant value defined by the SPI_URDATA register<br>01: Slave send the data frame received from master lastly<br>10: Slave send the data frame which is lastly transmitted by itself. (This data frame is stored in its TxFIFO)<br>11: Reserved |
| 8:5 | FIFOLVL[3:0] | FIFO threshold level<br>These bits show the number of data frames in a single packet. The size of the packet should not exceed half of the FIFO space.<br>0000: 1-data frame<br>0001: 2-data frame<br>0010: 3-data frame<br>0011: 4-data frame<br>….<br>1101: 14-data frame<br>1110: 15-data frame |

1111: 16-data frame

The SPI interface is more efficient if the configured packet size is aligned with the parallel bits of data register access. It is best to choose FIFOLVL from 2, 4, 6 etc when DZ <= 8 bits and the SPI data register is accessed by half-word. If DZ > 8 bits and the SPI data register is accessed by word, it is best to choose FIFOLVL from 2, 4, 6 etc. While if DZ <= 8 bits, it is best to choose FIFOLVL from 4, 8, 12 etc.

| | | |
|---|---|---|
| 4:0 | DZ[4:0] | Date size<br>These bits configure the number of bits in a data frame:<br>00000: Reserved<br>00001: Reserved<br>00010: Reserved<br>00011: 4-bit (When data width is 4 bit, must use the word / half word access FIFO, otherwise there is a risk of data disorder)<br>00100: 5-bit<br>00101: 6-bit<br>00110: 7-bit<br>……<br>11101: 30-bit<br>11110: 31-bit<br>11111: 32-bit |

## 27.5.4. Configuration register 1 (SPI_CFG1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by word (32-bit). When SPI is enable or IOAFEN bit is set to 1, this register is write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AFCTL | NSSCTL | NSSDRV | NSSIOPL | Reserved | NSSIM | CKPL | CKPH | LF | MSTMOD | TMOD | Reserved | | BDEN | BDOEN | RO |
| rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | | | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SWPMIO | Reserved | | | | | | | MDFD[3:0] | | | | | MSSD[3:0] | | |
| rw | | | | | | | | rw | | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | AFCTL | AF GPIOs control<br>This bit can be set or cleared when SPI is disable.<br>0: Peripherals do not control GPIO pins when disabled<br>1: Peripherals always control all associated GPIO pins<br>When the SPI master must be temporarily disabled for specific configuration reasons (such as CRC reset, or CKPH change), setting this bit to 1 forces the relevant output configured for alternate function mode to the state corresponding to |

| | | |
|---|---|---|
| | | the current SPI configuration to prevent an exit burr. In slave mode, this bit should not be used, because any slave transmitter cannot enforce its MISO output once the SPI is disabled.<br>**NOTE:** AFCTL is prohibited in MFD, MRU, and MRB modes. |
| 30 | NSSCTL | NSS pin output control in master mode<br>0: The NSS remains active level until the data transfer is complete, after that becomes inactive level via the ET flag<br>1: When MDFD[3:0] > 1, SPI data frames are interleaved with NSS invalid pulses |
| 29 | NSSDRV | NSS pin output enable in master mode<br>0: Output disable<br>1: Output enable |
| 28 | NSSIOPL | NSS pin input/output polarity selection<br>0: Low level is active<br>1: High level is active |
| 27 | Reserved | Must be kept at reset value. |
| 26 | NSSIM | NSS input signal manage mode<br>0: The NSS PAD decides the NSS input value<br>1: The NSSI bit of SPI_CTL0 decides the NSS input value |
| 25 | CKPL | Clock polarity selection<br>0: CLK pin is pulled low when SPI is in idle status<br>1: CLK pin is pulled high when SPI is in idle status |
| 24 | CKPH | Clock phase selection<br>0: Capture the first data at the first clock transition<br>1: Capture the first data at the second clock transition |
| 23 | LF | LSB first mode<br>0: Transmit MSB first<br>1: Transmit LSB first<br>This bit has no meaning in SPI TI mode. |
| 22 | MSTMOD | Master mode enable<br>0: Slave mode<br>1: Master mode |
| 21 | TMOD | SPI TI mode enable<br>0: SPI TI mode disabled<br>1: SPI TI mode enabled |
| 20:19 | Reserved | Must be kept at reset value. |
| 18 | BDEN | Bidirectional enable<br>0: 2 line unidirectional transmit mode<br>1: 1 line bidirectional transmit mode. The information transfers between the MOSI |

| | | |
|---|---|---|
| | | pin in master and the MISO pin in slave |
| 17 | BDOEN | Bidirectional transmit output enable<br>When BDEN is set, this bit determines the direction of transfer.<br>0: Work in receive-only mode<br>1: Work in transmit-only mode |
| 16 | RO | Receive only<br>When BDEN is cleared, this bit determines the direction of transfer.<br>0: Full-duplex mode<br>1: Receive-only mode |
| 15 | SWPMIO | MOSI and MISO pin swap<br>0: No swap<br>1: Swap<br>When this bit is set to 1, MISO and MOSI pin alternate functions are switched. |
| 14:8 | Reserved | Must be kept at reset value. |
| 7:4 | MDFD[3:0] | Delay between the data frames in SPI master mode<br>0000: No delay<br>0001: 1 clock cycle delay<br>….<br>1111: 15 clock cycle delay<br>This bit has no meaning in SPI TI mode. |
| 3:0 | MSSD[3:0] | Delay between active edge of NSS and start transfer or receive data in SPI master mode<br>0000: No delay<br>0001: 1 clock cycle delay<br>….<br>1111: 15 clock cycle delay<br>This bit has no meaning in SPI TI mode. |

## 27.5.5. Interrupt register (SPI_INT)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | TXSERFIE | CONFEIE | FEIE | CRCERIE | RXOREIE | TXUREIE | TXFIE | ESTCIE | DPIE | TPIE | RPIE |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

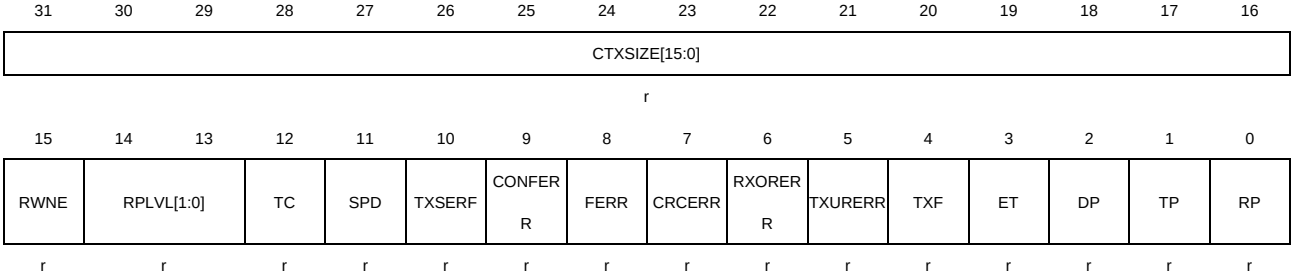
| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10 | TXSERFIE | TXSER reload interrupt enable<br>0: TXSER interrupt disable<br>1: TXSER interrupt enable |
| 9 | CONFEIE | SPI configuration error interrupt enable<br>0: SPI configuration error interrupt disable<br>1: SPI configuration error interrupt enable |
| 8 | FEIE | TI frame error interrupt enable<br>0: TI frame error interrupt disable<br>1: TI frame error interrupt enable |
| 7 | CRCERIE | CRC error interrupt enable<br>0: CRC error interrupt disable<br>1: CRC error interrupt enable |
| 6 | RXOREIE | Overrun error interrupt enable<br>0: Overrun interrupt disable<br>1: Overrun interrupt enable |
| 5 | TXUREIE | Underrun error interrupt enable<br>0: Underrun interrupt disable<br>1: Underrun interrupt enable |
| 4 | TXFIE | Transmission filled interrupt enable<br>0: TXF interrupt disable<br>1: TXF interrupt enable |
| 3 | ESTCIE | End of transfer or suspend or TxFIFO clear interrupt enable<br>0: ESTC interrupt disable<br>1: ESTC interrupt enable |
| 2 | DPIE | DP interrupt enable<br>0: DP interrupt disable<br>1: DP interrupt enable<br>This bit can be set by software, and will be cleared when TXF is set to 1. |
| 1 | TPIE | TP interrupt enable<br>0: TP interrupt disable<br>1: TP interrupt enable<br>This bit can be set by software, and will be cleared when TXF is set to 1. |
| 0 | RPIE | RP interrupt enable<br>0: RP interrupt disable<br>1: RP interrupt enable |

### 27.5.6. Status register (SPI_STAT)

Address offset: 0x14

Reset value: 0x0000 1002

This register can be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | CTXSIZE[15:0] | These bits show the number of data frames remaining in the TXSIZE bit feild (in the SPI_CTL1 register).<br>This value is not very reliable when there is data transmission on the bus. |
| 15 | RWNE | The word of RxFIFO is not empty<br>0: Data takes RxFIFO space less than a word (32-bit)<br>1: Data takes RxFIFO space at least a word (32-bit) |
| 14:13 | RPLVL[1:0] | RxFIFO packet level<br>These bits define the number of data frames stored in the last 32-bit word area of RxFIFO.<br>If the size of data frame <= 8 bit (DZ[4:0] <= 7):<br>00: there are no frame (RWNE = 0) or multiple of 4 farmes (RWNE = 1) stored in RxFIFO<br>01: there is 1 frame stored in RxFIFO (RWNE = 0)<br>10: there are 2 frames stored in RxFIFO (RWNE = 0)<br>11: there are 3 frames stored in RxFIFO (RWNE = 0)<br>If the size of data frame > 8 bit and <= 16 bit (7 < DZ[4:0] <= 15):<br>00: there are no frame (RWNE = 0) or multiple of 2 farmes (RWNE = 1) stored in RxFIFO<br>01: there is 1 frame stored in RxFIFO (RWNE = 0)<br>Other: not used.<br>If the size of data frame > 16 bit (DZ[4:0] > 15):<br>00: only read<br>Other: not used. |
| 12 | TC | TxFIFO transmission complete flag<br>0: There are data stored in TxFIFO, or the last frame is sent ongoing (including CRC)<br>1: The last data or CRC frame is sent complete |

| | | |
|---|---|---|
| 11 | SPD | Suspend flag<br>0: SPI is not suspended<br>1: SPI master mode is suspended |
| 10 | TXSERF | The additional SPI data has been reloaded<br>0: no data accepted<br>1: The current transaction continues after receiving an additional number of data<br>This bit can be cleared by write 1 to TXSERFC bit (in SPI_STATC register) or program TXSER bits (in SPI_CTL1 register). |
| 9 | CONFERR | SPI configuration error<br>0: No configuration error<br>1: Configuration error occurred |
| 8 | FERR | SPI TI format error<br>0: No TI format error<br>1: TI format error occurs |
| 7 | CRCERR | SPI CRC error<br>0: No CRC error<br>1: CRC error occurs |
| 6 | RXORERR | Reception overrun error<br>0: No reception overrun error<br>1: Reception overrun error occurs |
| 5 | TXURERR | Transmission underrun error<br>0: No transmission underrun error<br>1: Transmission underrun error occurred |
| 4 | TXF | TxFIFO transmission has been filled<br>0: TxFIFO upload ongoing or not start<br>1: TxFIFO upload complete |
| 3 | ET | End of transmission / reception flag<br>0: Transmission / reception ongoing or not start<br>1: Transmission / reception complete |
| 2 | DP | Duplex packet<br>0: TxFIFO is full and / or RxFIFO is empty<br>1: TxFIFO has space to write a complete data frame (TP = 1) and RxFIFO contains at least one packet to read (RP = 1) |
| 1 | TP | TxFIFO packet space available flag<br>0: TxFIFO don't have enough space to receive the next packet<br>1: TxFIFO have enough available space to receive a packet |
| 0 | RP | RxFIFO packet space available flag<br>0: The RxFIFO is empty or the received packet is incomplete (not reach at FIFOLVL) |

1: RxFIFO contains at least one complete packet

### 27.5.7. Interrupt/Status flags clear register (SPI_STATC)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | SPDC | TXSERFC | CONFERRC | FERRC | CRCERRC | RXORERRC | TXURERRC | TXFC | ETC | Reserved | | |
| | | | | w | w | w | w | w | w | w | w | w | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | SPDC | Clear the suspend flag<br>Write 1 to this bit can clear the SPD bit in the SPI_STAT register. |
| 10 | TXSERFC | Clear the TXSERF flag<br>Write 1 to this bit can clear the TXSERF bit in the SPI_STAT register. |
| 9 | CONFERRC | Clear the configuration error flag<br>Write 1 to this bit can clear the CONFERR bit in the SPI_STAT register. |
| 8 | FERRC | Clear the SPI TI format error flag<br>Write 1 to this bit can clear the FERR bit in the SPI_STAT register. |
| 7 | CRCERRC | Clear the CRC error flag<br>Write 1 to this bit can clear the CRCERR bit in the SPI_STAT register. |
| 6 | RXORERRC | Clear the reception overrun error flag<br>Write 1 to this bit can clear the RXORERR bit in the SPI_STAT register. |
| 5 | TXURERRC | Clear the transmission underrun error flag<br>Write 1 to this bit can clear the TXURERR bit in the SPI_STAT register. |
| 4 | TXFC | Clear the TxFIFO transmission filled flag<br>Write 1 to this bit can clear the TXF bit in the SPI_STAT register. |
| 3 | ETC | Clear the end of transmission/reception flag<br>Write 1 to this bit can clear the ET bit in the SPI_STAT register. |
| 2:0 | Reserved | Must be kept at reset value. |

### 27.5.8. Data Transfer register (SPI_TDATA)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TDATA[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TDATA[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TDATA[31:0] | Data transfer register<br>The hardware has two FIFOs, including TxFIFO and RxFIFO. The SPI_TDATA register serves as an interface with TxFIFO. Write data to SPI_TDATA will save the data to TxFIFO. Data is always right-aligned, and according to WORDEN, BYTEN, DZ value to place data. For example: if WORDEN is set 1, DZ is 8-bit, TDATA [7:0] is data 0, TDATA [15:8] is data 1, TDATA [23:16] is data 2, TDATA [31:24] is data 3. If WORDEN is set 0, BYTEN is set 0, DZ is 8-bit, TDATA [7:0] is data 0, TDATA [15:8] is data 1, TDATA [31:16] is invalid data. If WORDEN is set 0, BYTEN is set 1, DZ is 8-bit, TDATA [7:0] is data 0, TDATA [31:8] is invalid data. If DZ is more than 8-bit, only according to the word or half word access. |

### 27.5.9. Data Receive register (SPI_RDATA)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RDATA[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RDATA[31:0] | Data receive register<br>The hardware has two FIFOs, including TxFIFO and RxFIFO. The SPI_RDATA register serves as an interface with RxFIFO. Read data from SPI_RDATA will get the data from RxFIFO. Data is always right-aligned, and according to WORDEN, BYTEN, DZ value to place data. For example: if WORDEN is set 1, DZ is 8-bit, |

RDATA [7:0] is data 3, RDATA [15:8] is data 2, RDATA [23:16] is data 1, RDATA [31:24] is data 0. If WORDEN is set 0, BYTEN is set 0, DZ is 8-bit, RDATA [15:0] is invalid data , RDATA [23:16] is data 1, RDATA [31:24] is data 0. If WORDEN is set 0, BYTEN is set 1, DZ is 8-bit, RDATA [23:0] is invalid data , RDATA [31:24] is data 0. If DZ is more than 8-bit, only according to the word or half word access.

## 27.5.10. CRC polynomial register (SPI_CRCPOLY)

Address offset: 0x40
Reset value: 0x0000 0107

This register can be accessed by word (32-bit). When SPI is enable, this register is write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRCPOLY[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRCPOLY[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CRCPOLY[31:0] | CRC polynomial register<br>This register contains the CRC polynomial and it is used for CRC calculation.<br>The default value of 0x107 is corresponding to DZ = 8-bit. It is compatible with some other GD products with fixed length polynomial string (use the default value 0x07).<br>The polynomial length is defined by the most significant bit of SPI_CRCPOLY register. The length value should be greater than the DZ value. In addition, if DZ = 32-bit, the CRCFS must be set to 1 to make sure the size of polynomial greater than the size of data.<br>If DZ = 16-bit, the 16-31 bit fields of SPI_CRCPOLY register should be reserved. When the register is accessed by word (32-bit), the 16-31 bit fields always read zero, and write invalid. |

## 27.5.11. TX CRC register (SPI_TCRC)

Address offset: 0x44
Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCRC[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCRC[15:0] | | | | | | | | | | | | | | | |

r

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TCRC[31:0] | Tx CRC register.<br>When the CRCEN bit of SPI_CFG0 is set, the hardware computes the CRC value of the transmitted bytes and saves them in SPI_TCRC register.<br>These bits have no meaning in I2S mode. |

## 27.5.12. RX CRC register (SPI_RCRC)

Address offset: 0x48
Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RCRC[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RCRC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RCRC[31:0] | Rx CRC register.<br>When the CRCEN bit of SPI_CFG0 is set, the hardware computes the CRC value of the received bytes and saves them in SPI_RCRC register.<br>These bits have no meaning in I2S mode. |

## 27.5.13. Underrun Data register (SPI_URDATA)

Address offset: 0x4C
Reset value: 0x0000 0000

This register can be accessed by word (32-bit). When SPI is enable, this register is write protected.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| URDATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| URDATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | URDATA[31:0] | Transmission underrun data at slave mode.<br>This register is considered only in slave mode and underrun conditions. The number of bits considered depends on the DZ bits setting of SPI_CFG0 register. The |

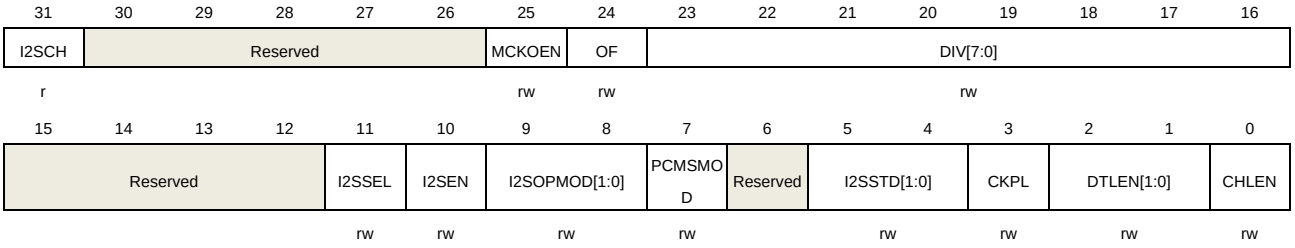
treatment of the underrun condition depends on the TXURDT and TXUROP bits of the SPI_CFG0 register.

### 27.5.14. I2S control register (SPI_I2SCTL)

Address offset: 0x50

Reset value: 0x0000 0000

This register can be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | I2SCH | I2S channel side<br>0: The next data needs to be transmitted or the data just received is channel left.<br>1: The next data needs to be transmitted or the data just received is channel right.<br>This bit is set and cleared by hardware.<br>This bit is not used in SPI mode, and has no meaning in the I2S PCM mode.<br>For TX, this bit is meaningful only if FIFOLVL = 15, that is, TxFIFO is only used for sending 1 data.<br>For RX, this bit is meaningful only if FIFOLVL = 0, that is, RXFIFO is used for receiving only one data.<br>For other configurations, the bit has no clear meaning. |
| 30:26 | Reserved | Must be kept at reset value. |
| 25 | MCKOEN | I2S_MCK output enable<br>0: I2S_MCK output is disabled<br>1: I2S_MCK output is enabled<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 24 | OF | Odd factor for the prescaler<br>0: Real divider value is DIV * 2<br>1: Real divider value is DIV * 2 + 1<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 23:16 | DIV[7:0] | Dividing factor for the prescaler<br>Real divider value is DIV * 2 + OF.<br>DIV must not be 0.<br>These bits should be configured when I2S mode is disabled. |

| | | |
|---|---|---|
| | | These bits are not used in SPI mode. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11 | I2SSEL | I2S mode selection<br>0: SPI mode<br>1: I2S mode<br>This bit should be configured when SPI/I2S is disabled. |
| 10 | I2SEN | I2S enable<br>0: I2S is disabled<br>1: I2S is enabled<br>This bit is not used in SPI mode. |
| 9:8 | I2SOPMOD[1:0] | I2S operation mode<br>00: Slave transmission mode<br>01: Slave reception mode<br>10: Master transmission mode<br>11: Master reception mode<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 7 | PCMSMOD | PCM frame synchronization mode<br>0: Short frame synchronization<br>1: long frame synchronization<br>This bit has a meaning only when PCM standard is used.<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 6 | Reserved | Must be kept at reset value. |
| 5:4 | I2SSTD[1:0] | I2S standard selection<br>00: I2S Philips standard<br>01: MSB justified standard<br>10: LSB justified standard<br>11: PCM standard<br>These bits should be configured when I2S mode is disabled.<br>These bits are not used in SPI mode. |
| 3 | CKPL | Idle state clock polarity<br>0: The idle state of I2S_CK is low level<br>1: The idle state of I2S_CK is high level<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |
| 2:1 | DTLEN[1:0] | Data length<br>00: 16-bit<br>01: 24-bit<br>10: 32-bit |

| | | |
|---|---|---|
| | | 11: Reserved<br>These bits should be configured when I2S mode is disabled.<br>These bits are not used in SPI mode. |
| 0 | CHLEN | Channel length<br>0: 16-bit<br>1: 32-bit<br>The channel length must be equal to or greater than the data length.<br>This bit should be configured when I2S mode is disabled.<br>This bit is not used in SPI mode. |

## 27.5.15. Quad_SPI mode control register (SPI_QCTL)

Address offset: 0x80
Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | QRD | QMOD |
| | | | | | | | | | | | | | | rw | rw |

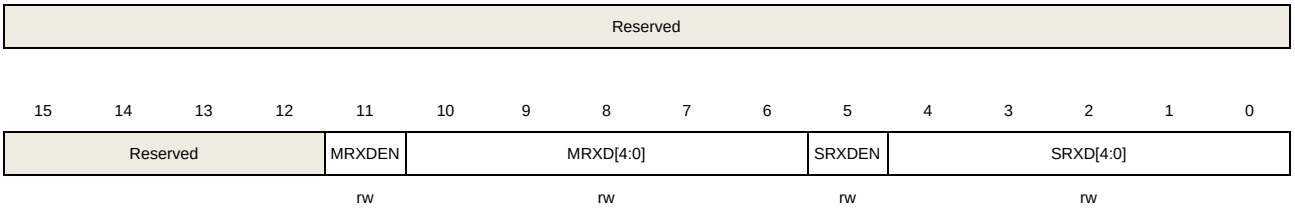
| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value |
| 1 | QRD | Quad-SPI mode read select<br>0: SPI is in quad wire write mode<br>1: SPI is in quad wire read mode<br>This bit should be only be configured when SPI is not busy.<br>This bit is only available in SPI3 / 4. |
| 0 | QMOD | Quad-SPI mode enable<br>0: SPI is in single wire mode<br>1: SPI is in Quad-SPI mode<br>This bit should only be configured when SPI is not busy.<br>This bit is only available in SPI3 / 4. |

## 27.5.16. RX clock delay register (SPI_RXDLYCK)

Address offset: 0xFC
Reset value: 0x0000 0000

This register can be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|




| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value |
| 11 | MRXDEN | When master receive, sampling clock delay enable<br>0: Sampling clock delay enable<br>1: Sampling clock delay disable |
| 10:6 | MRXD[4:0] | When master receive, sampling clock delay units<br>00000: Delay 1 unit<br>00001: Delay 2 units<br>…<br>11111: Delay 32 units |
| 5 | SRXDEN | When slave receive, sampling clock delay enable<br>0: Sampling clock delay enable<br>1: Sampling clock delay disable |
| 4:0 | SRXD[4:0] | When slave receive, sampling clock delay units<br>00000: Delay 1 unit<br>00001: Delay 2 units<br>…<br>11111: Delay 32 units |

# 28. OSPI I/O manager(OSPIM)

## 28.1. Overview

OSPIM supports OSPI pin assignment with full matrix.

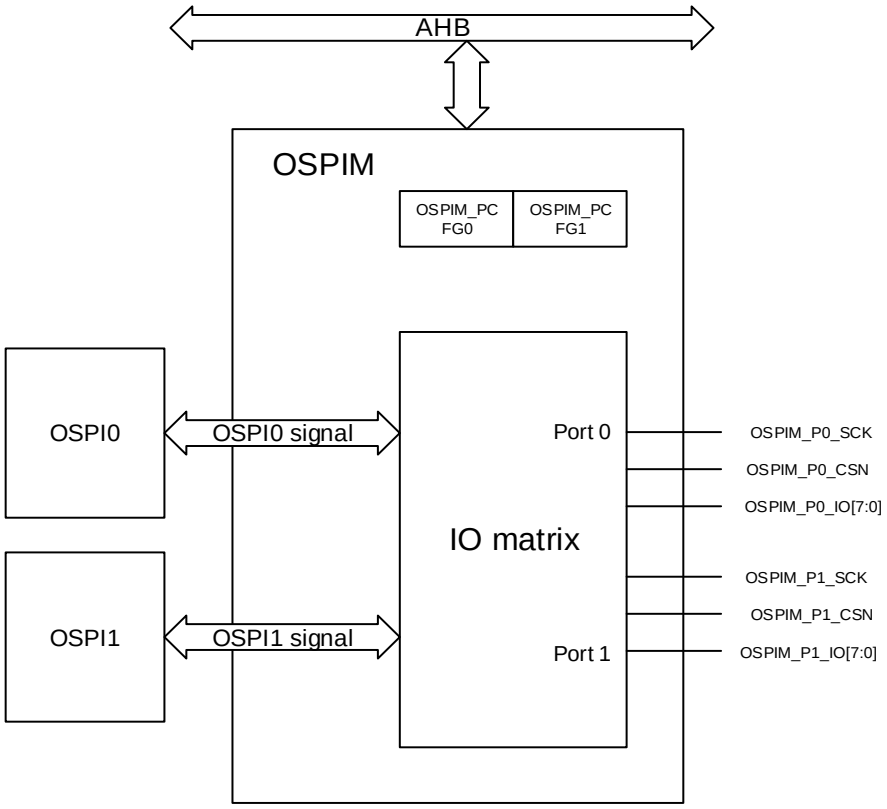
## 28.2. Characteristics

- Support two SPI(single-line, two-lines, four-lines, eight-lines) interfaces.
- Support two ports for pin assignment.
- Fully programmable IO matrix, can assign pins according to function.

**Note**: For GD32H75E, OSPI1 and OSPIM Port1 is only for internal ESC and cannot be used for external devices.

## 28.3. Function overview

### 28.3.1. OSPIM block diagram



### 28.3.2. OSPIM matrix

The OSPIM matrix is fully programmable, and the pins can be pre-mapped according to function, as shown in the ***Table 28-1 OSPIM matrix mapping***:

**Table 28-1 OSPIM matrix mapping**

| Pins | Mapping |
|---|---|
| OSPIM_P0_SCK, OSPIM_P1_SCK | Can be independently mapped to OSPI0_SCK or OSPI1_SCK |
| OSPIM_P0_DQS, OSPIM_P1_DQS | Can be independently mapped to OSPI0_DQS or OSPI1_DQS |
| OSPIM_P0_CSN, OSPIM_P1_CSN | Can be independently mapped to OSPI0_CSN or OSPI1_CSN |
| OSPIM_P0_IO[3:0], OSPIM_P0_IO[7:4], OSPIM_P1_IO[3:0], OSPIM_P1_IO[7:4] | Can be independently mapped to OSPIM0_IO[3:0], OSPIM0_IO[7:4], OSPIM1_IO[3:0] or OSPIM1_IO[7:4] |

By default, the signals of OSPI0 and OSPI1 are mapped to port 0 and port 1, respectively. Port 0 and port 1 of OSPIM can be independently configured through OSPIM_PCFGx register. If OSPIs are disabled, the OSPIM matrix must be configured to avoid unexpected transactions on the bus.

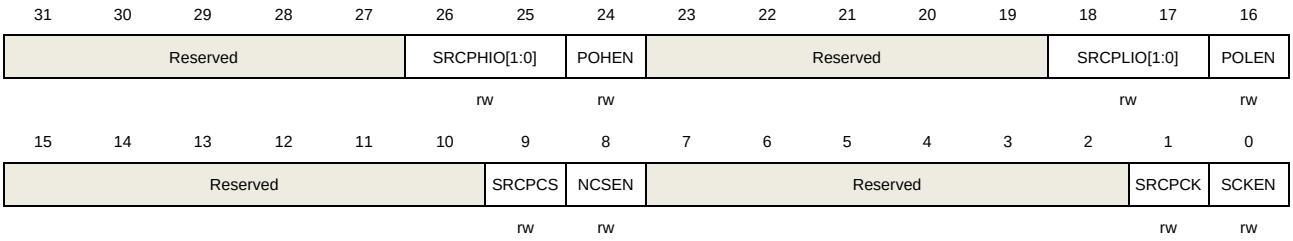
## 28.4. Register definition

OSPIM base address: 0x5200 B400

### 28.4.1. Port configuration register (OSPIM_PCFGx) (x = 0, 1)

Address offset: 0x04*(x+1)
Reset value: 0x0301 0111 (x = 0), 0x0705 0333 (x=1)

This register can be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26:25 | SRCPHIO[1:0] | Source selection for IO[7:4] of port x<br>00: Select OSPI0_IO[3:0].<br>01: Select OSPI0_IO[7:4]. |

| | | |
|---|---|---|
| | | 10: Select OSPI1_IO[3:0].<br>11: Select OSPI1_IO[7:4]. |
| 24 | POHEN | Enable for IO[7:4] of port x<br>0: Disable IO[7:4] of port x.<br>1: Enable IO[7:4] of port x. |
| 23:19 | Reserved | Must be kept at reset value. |
| 18:17 | SRCPLIO[1:0] | Source selection for IO[3:0] of port x<br>00: Select OSPI0_IO[3:0].<br>01: Select OSPI0_IO[7:4].<br>10: Select OSPI1_IO[3:0].<br>11: Select OSPI1_IO[7:4]. |
| 16 | POLEN | Enable for IO[3:0] of port x<br>0: Disable IO[3:0] of port x.<br>1: Enable IO[3:0] of port x. |
| 15:10 | Reserved | Must be kept at reset value. |
| 9 | SRCPCS | Source selection for CSN of port x<br>0: The source of CSN is OSPI0_CSN<br>1: The source of CSN is OSPI1_CSN |
| 8 | NCSEN | Enable for CSN of port x<br>0: Disable CSN of port x<br>1: Enable CSN of port x |
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | SRCPCK | Source selection for SCK of port x<br>0: The source of SCK is OSPI0_SCK<br>1: The source of SCK is OSPI1_SCK |
| 0 | SCKEN | Enable for SCK of port x<br>0: Disable SCK of port x<br>1: Enable SCK of port x |

# 29. Octal-SPI interface(OSPI)

## 29.1. Overview

The OSPI is a specialized interface that communicate with external memories. The interface support single, dual, quad and octal SPI mode.

## 29.2. Characteristics

- Three functional modes:
  - indirect mode: all operations are performed depends on OSPI registers.
  - status polling mode: the values of status registers in external memory are periodically read and check.
  - memory-mapped mode: the external memory is mapped to the microcontroller address space(OSPI0: 0x9000 0000 – 0x9FFF FFFF, OSPI1: 0x7000 0000 – 0x7FFF FFFF) and is accessed as an internal memory.
- Support read in memory-mapped mode.
- Support single, dual, quad and octal communication.
- Fully programmable command format for both indirect and memory-mapped mode.
- Support SDR(signal data rate) and DTR(double transfer rate, only for GD25LX512ME read).
- Integrated FIFO for transmission/reception.
- 8, 16 and 32-bits data access.
- DMA channel for indirect mode.
- Interrupt generation on FIFO threshold, status match, transfer complete and access error.

**Note**: For GD32H75E, OSPI1 and OSPIM Port1 is only for internal ESC and cannot be used for external devices.

## 29.3. Functon overview

### 29.3.1. OSPI block diagram

The pins of OSPI are described in the table below.

**Table 29-1 OSPI signal description**

| Pin name | Direction | Description |
|---|---|---|
| CSN | O | chip select output(active low) |
| SCK | O | clock output |
| IO0/SO | I/O | single mode: data output.<br>dual mode: data input or output.<br>qual mode: data input or output. |

| Pin name | Direction | Description |
|---|---|---|
| | | octal mode: data input or output. |
| IO1/SI | I/O | single mode: data intput.<br>dual mode: data input or output.<br>qual mode: data input or output.<br>octal mode: data input or output. |
| IO2 | I/O | single mode: ouptput mode and forced to 0, connect WP pin of external memories, control “write protect” function.<br>dual mode: ouptput mode and forced to 0, connect WP pin of external memories, control “write protect” function.<br>quad mode: data input or output.<br>octal mode: data input or output. |
| IO3 | I/O | single mode: output mode and forced to 1, connect HOLD pin of external memories, control “hold” function.<br>dual mode: output mode and forced to 1, connect HOLD pin of external memories, control “hold” function.<br>quad mode: data input or output.<br>octal mode: data input or output. |
| IO4 | I/O | single mode: output mode and forced to 0.<br>dual mode: output mode and forced to 0.<br>quad mode: output mode and forced to 0.<br>octal mode: data input or output. |
| IO5 | I/O | single mode: output mode and forced to 0.<br>dual mode: output mode and forced to 0.<br>quad mode: output mode and forced to 0.<br>octal mode: data input or output. |
| IO6 | I/O | single mode: output mode and forced to 0.<br>dual mode: output mode and forced to 0.<br>quad mode: output mode and forced to 0.<br>octal mode: data input or output. |
| IO7 | I/O | single mode: output mode and forced to 0.<br>dual mode: output mode and forced to 0.<br>quad mode: output mode and forced to 0.<br>octal mode: data input or output. |

**Figure 29-1 OSPI octal communication mode block diagram**

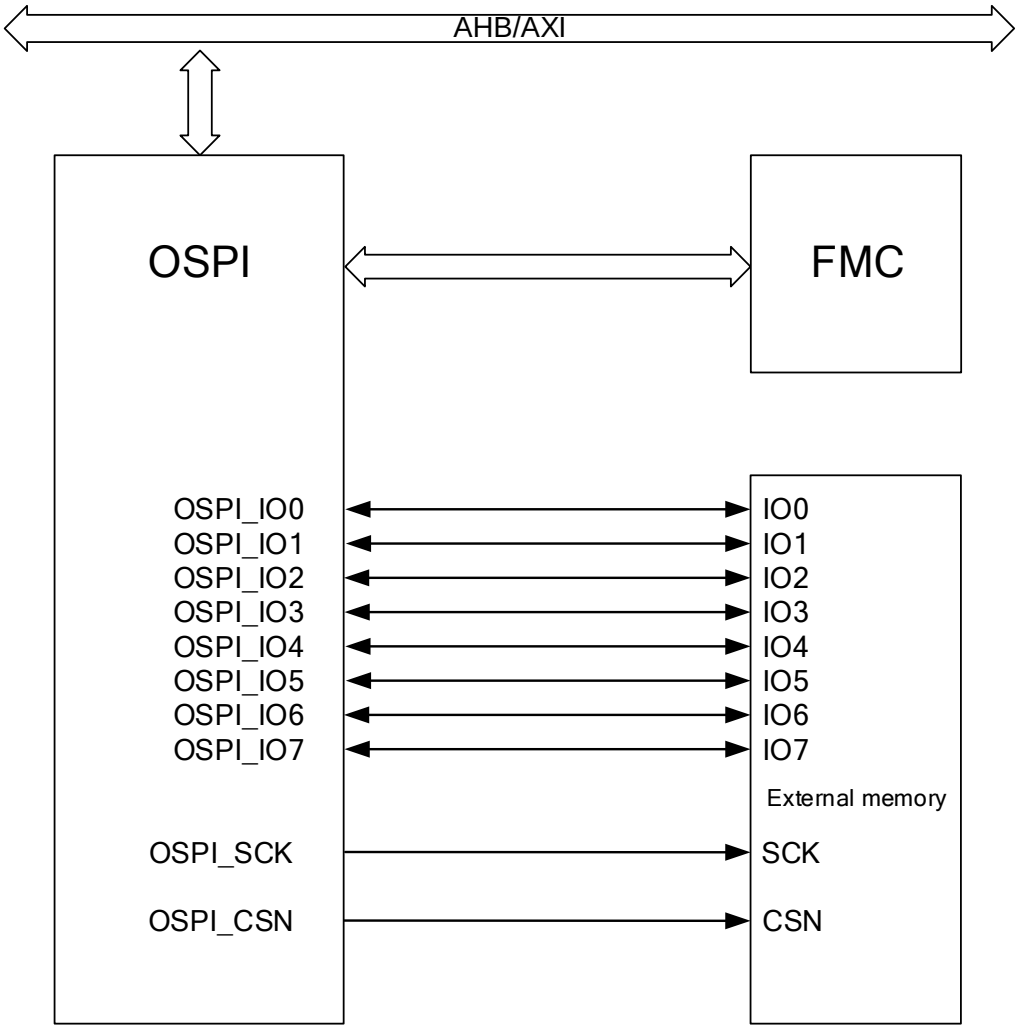
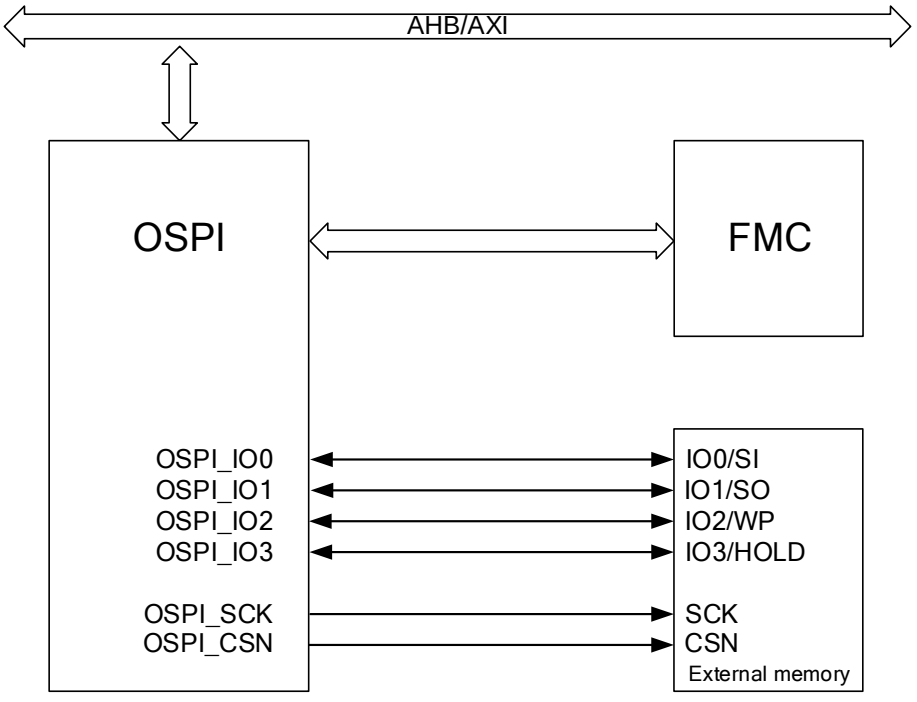



**Figure 29-2 OSPI quad communication mode block diagram**



### 29.3.2. OSPI regular command format

In regular command, there are totally 5 phases which can be included or not: instruction, address, alternate-bytes, dummy and data. Any of these phases can be omitted or not, but at least one of the instructions phase, address phase, alternate byte phase and data phase must be present, this must be guaranteed by software, hardware is not designed to provide any protection methods. In addition, the most-signification-bit always occupies the highest IO line number.

**Figure 29-3 OSPI command format in octal mode**

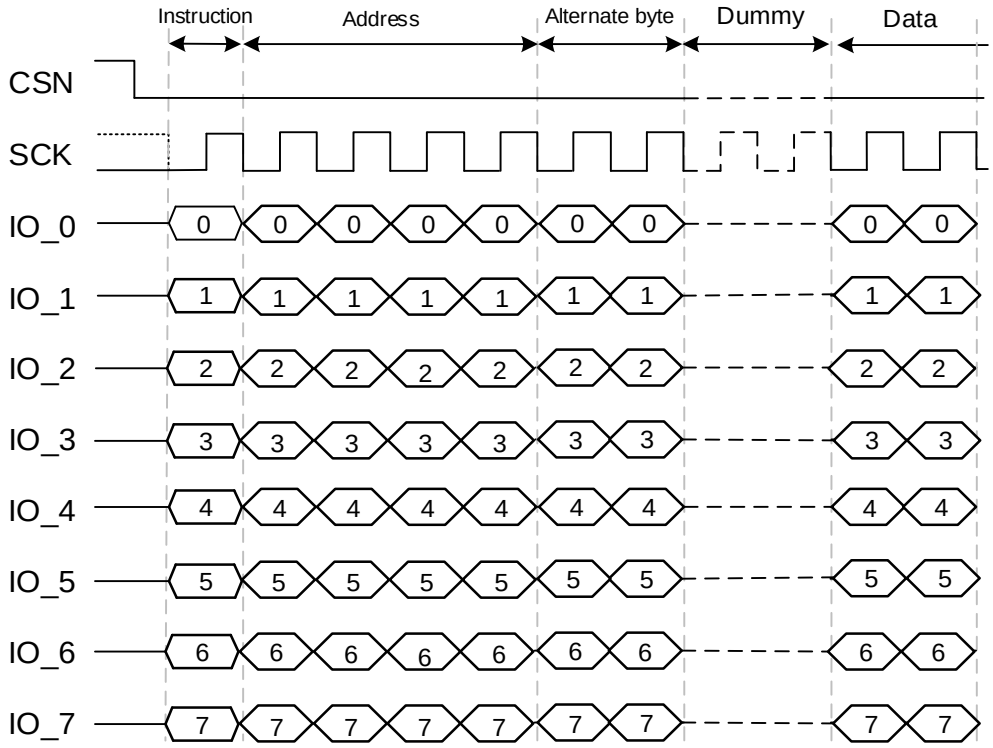
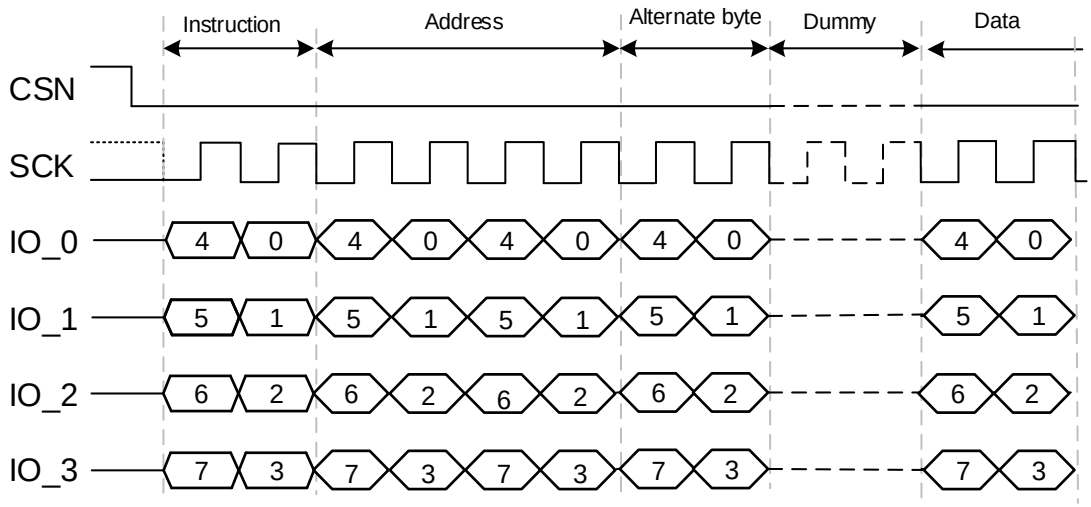



**Figure 29-4 OSPI command format in quad mode**



### Instruction phase

In instruction phase, configure the command size to be sent in the INSSZ[1:0] bit field of the OSPI_TCFG register(8, 16, 24 or 32 bits), configure the instructions to be sent to the external memory in the OSPI_INS register. The IMOD field of the OSPI_TCFG register defines the instruction phase mode (no instruction, 1-line, 2-lines, 4-lines or 8-lines).

When FMOD[1:0] bit field of the OSPI_CTL register is 0b11, the OSPI works in memory-mapped mode. In memory-mapped mode, define the write operation instruction in the OSPI_WINS register, configure the format of the instruction in the OSPI_WTCFG register (size and mode). Configure the read operation instruction and the format of the instruction in the OSPI_INS register and OSPI_TCFG register.

### Address phase

In address phase, configure the address size to be sent in the ADDRSZ[1:0] bit field of the OSPI_TCFG register(8, 16, 24 or 32 bits). The ADDRMOD[2:0] field of the OSPI_TCFG register defines the address phase mode (no address, 1-line, 2-lines, 4-lines or 8-lines). Set the ADDRDTR bit of the OSPI_TCFG register to enable the DTR mode, and the address is sent on each edge of the clock. In indirect mode and status polling mode, define the address information to be sent in the OSPI_ADDR register.

In memory-mapped mode, the address to be sent is directly given by AXI (Cotex-M core or DMA). Configure the format of the write operation address in the OSPI_WTCFG register (size, mode and whether to enable DTR). Configure the the format of read operation address in the OSPI_TCFG register.

### Alternate-bytes phase

In alternate-bytes phase, configure the alternate bytes size to be sent in ALTESZ[1:0] bit field of the OSPI_TCFG register(8, 16, 24 or 32 bits), configure the alternate bytes to be sent to the external memory in the OSPI_ALTE register. The ALTEMOD[2:0] field of the OSPI_TCFG register defines the alternate-bytes phase mode (no alternate-bytes, 1-line, 2-lines, 4-lines or 8-lines). Set the ALTEDTR bit of the OSPI_TCFG register to enable the DTR mode, and the alternate bytes is sent on each edge of the clock.

In memory-mapped mode, define the write operation alternate bytes in the OSPI_WALTE register, configure the format of the alternate bytes in the OSPI_WTCFG register (size, mode and whether to enable DTR). Configure the read operation alternate bytes and the format of the alternate bytes in the OSPI_ALTE register and OSPI_TCFG register.

### Dummy phase

In dummy pahse, 0-31 cycles, as specified by DUMYC[4:0] bit field in OSPI_TIMCFG register, are given without any data being transferred for external memory, in order to wait flash prepare data.

In memory-mapped mode, specify write operation dummy cycles in DUMYC[4:0] bit field of

the OSPI_WTIMCFG register, and specify read operation dummy cycles in DUMYC[4:0] bit field of the OSPI_TIMCFG register.

**Note:**

At least 1 dummy cycle when OSPI is working in 2-lines, 4-lines or 8-lines mode to receive data from external memory.

### Data phase

In data phase, any number of bytes can be transferred between the external memory and the OSPI interface. The DATAMOD[2:0] bit field of the OSPI_TCFG register defines the mode of the data phase(no data, 1-line, 2-lines, 4-lines or 8-lines, of which no data can only be used in indirect write mode). Set the DADTR bit of the OSPI_TCFG register to enable the DTR mode, and the data is sent on each edge of the clock. In indirect mode, the OSPI_DTLEN register defines the number of bytes to be sent or received. In write operation, data to be sent should be written to the OSPI_DATA register, while in read operation, received data is obtained by reading OSPI_DATA register. In status polling mode, OSPI_DTLEN register defines the number of bytes to be received, and the receive data is obtained from the OSPI_DATA register.

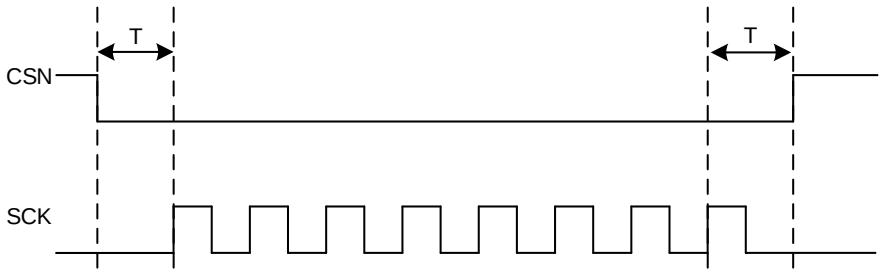
In memory-mapped mode, data to be sent is directly given by AXI (Cotex-M core or DMA). The number of bytes transmitted determines the access operation of the AXI bus, which can be read and written in 8-bit, 16-bit or 32-bit, and correspondingly transfer 1, 2, or 4 bytes. Configure the format of the data in the OSPI_WTCFG register(mode and whether to enable DTR) for sending. Configure the format of the data in the OSPI_TCFG register(mode and whether to enable DTR) for reading.

### CSN and SCK behavior

The default value of CSN is high, and it falls before a command begins and rises soon as it finishes. SCK out put signal is a gate signal from internal SCK, where the internal SCK is present all the time.

CSN falls one SCK cycle befor the first valid rising SCK edge, and rises on SCK cycle after the final valid rising SCK edge. Refer to ***Figure 29-5 CSN and SCK behavior***.

**Figure 29-5 CSN and SCK behavior**



## 29.4. Operating modes

### 29.4.1. Indirect mode

In indirect write mode, data to be transmitted are written in OSPI_DATA register. While in indirect read mode, data to be received are read from OSPI_DATA register.

OSPI_DTLEN register defines the number of byte to be transferred. If DTLEN = 0xFFFF FFFF, the number of data is considered undefined, the transmission continues until the memory size boundary is reach as sepcifiled by MESZ[4:0]. If both DTLEN = 0xFFFF FFFF and MESZ[4:0] = 0x1F, then the transmission continues indefinitely until the OSPI is disabled.

Transfer complete flag TC is set when the number of byte programed in DTLEN is reached, in case of undefined transfer length, TC is set when the transmit/received byte number equals to external memory size. An interrupt is generated if TCIE and TC are both set, and it is cleared by setting TCC to 1.

#### Trigger a command sequence in regular command mode

The command sequence starts immediately after the last information is provided by software according to communication requirement. After the command starts, the BUSY bit is set to 1.

When neither address nor data are required, the sequence start immediately after OSPI_INS has be accessed.

When address is required and no data is required, the sequence starts after OSPI_ADDR has been accessed.

When both address and data are required in indirect write mode, the command sequence starts after OSPI_DATA has been accessed.

#### FIFO and flag control

A FIFO with a size of 8-bit by 32 is implemented to transfer data. In indirect write mode, 32-bit write access add 4-bytes to FIFO, 16-bit write access add 2-bytes to FIFO, and 8-bit add 1-byte.

FIFO threshold is defined by FTL, in indirect read mode, when the amount of bytes in the FIFO is equal or above the defined threshold, FIFO threshold flag FT is set. FT is also set after data phase is complete if FIFO is not empty. In indirect write mode, when the amount of the empty bytes in the FIFO is above the threshold, FT is set.

An interrupt is generated if both FTIE and FT is set. If DMA is enabled, a DMA request is generated by FT, until this flag is cleared.

In indirect read mode, when the FIFO becomes full, the OSPI temporarily stop SCK clock to avoid overrun. The reading sequence is not resumed until more than 4 byte are available in FIFO.

### 29.4.2. Status polling mode

In status polling mode, the OSPI periodically starts a read command with up-to 4-bytes data. If the data length defined in the OSPI_DTLEN register is greater than 4, ignore the extra length and only read 4 bytes. The received data can be bit-wise masked and compared with a defined data content, if a match happens, then an interrupt is generated when SMIE is set.

Status polling access starts the same as indirect read sequence. BUSY stays high even between periodic intervals.

Polling match mode SPMOD controls the comparison match mode, if SPMOD = 0, the AND mode is selected. In this mode, status match flag SM is set only when there is a match on all the unmasked bits. While if SPMOD = 1, the OR mode is selected. In this mode, SM is set if there is a match on any of the unmasked bit.

If status-polling-mode-stop SPS is set, status polling sequence stop when a match is detected, and the BUSY flag is cleared at the end of data phase.

In status polling mode, FIFO is bypassed, the read status bytes are stored in OSPI_DATA register, and the stored status bytes are not affected by the MASK control field. OSPI_DATA contents is renewed at the beginning of data phase.

FT is set at the end of data phase, where the external flash memory status bytes are considered read, and it is cleared when OSPI_DATA register is read.

In status polling mode, the external memory must be configured in fixed latency mode.

### 29.4.3. Memory map mode

In memory-mapped mode, the external flash memory is considered as internal memory, no more than 256MB can be address even if the external memory is larger. The Memory map mode also don’t allow an an address outside what defined by MESZ[4:0] but still within 256MB range.

If any of above condition happens, AXI will generate an error .The effect of the error depends on the AXI master. If the master is CPU, a hartfault is generate, if it is DMA, transfer error is generated, and the corresponding DMA channel is disabled.

In this mode, byte, half-word, and word single or burst access are supported.

Execute in place (XIP) is also supported, where OSPI continue to load bytes to the address after completing the most recent access. If the subsequent access is indeed made at a continuous address, the access with be completed faster since the value is already prefetched. Otherwise, the read sequence is restarted, polling CSN low before the read sequence starts.

After the FIFO is empty, the OSPI enters hold state, in which no SCK is sent, CSN is maintained low during this period. At the beginning of a transfer, BUSY goes high before CSN falls.

## 29.5. OSPI configuration

### 29.5.1. OSPI system configuration

The details of the OSPI system configuration as follows:

1. Configure the functional mode of OSPI with FMOD[1:0] bit field.
2. If OSPI is in status polling mode, then need to configure the SPMOD and SPS bits to select the polling matching mode and the stop mode of the automatic polling mode.
3. Configure the threshold of FIFO with FTL[4:0] bit field.
4. If need to use DMA, set the DMAEN bit to 1. Must be disable DMA during OSPI configuration, otherwise unexpected requests may be generated.
5. If need to use interrupt, the respective interrupr enable bie can be set.

### 29.5.2. OSPI device configuration

Configure parameters of OSPI and external device by OSPI_DCFG0 and OSPI_DCFG1 register, the details as follows:

1. Configure the external flash memory type by set the value of DTYSEL[2:0] bit field.
2. Configure the size of external flash memory by set the value of MESZ[4:0][4:0].
3. Configure chip-select minimum high time by set the value of CSHC[5:0].
4. Configure prescaler by set the value of PSC[7:0].

### 29.5.3. OSPI regular commamd configuration

#### Indirection mode

The details of the OSPI system configuration as follows:

1. Configuration the data length by set OSPI_DTLEN register.
2. Configuration the frame timing by set OSPI_TIMCFG register.
3. Configuration the frame format by set OSPI_TCFG register.
4. Specify the instruction to be sent to the external flash memory by set OSPI_INS register.
5. Specify the alternate bytes to be sent to the external flash memory immediately after the address is sent by set OSPI_ALTE register.
6. Specify the address to be sent to the external flash memory by set OSPI_ADDR register.
7. Read/write the data through OSPI_DATA register.

#### Status polling mode

When OSPI is in status polling mode, the details as follows:

1. Specify to mask the received status byte by setting OSPI_STATMK register.
2. Specify the value to be compared with the OSPI_STATMK register by setting OSPI_STATMATCH.
3. Specify the number of clock cycles between read operations by setting OSPI_INTERVAL

register.

4. Configure the data length by setting OSPI_DTLEN register.
5. Configure the frame timing by setting OSPI_TIMCFG register.
6. Configure the frame format by setting OSPI_TCFG register.
7. Specify the instruction to be sent to the external flash memory by set OSPI_INS register.
8. Specify the alternate bytes to be sent to the external flash memory immediately after the address is sent by setting OSPI_ALTE register.
9. Specify the address to be sent to the external flash memory by set OSPI_ADDR register.

#### Memory map mode

In memory map mode, external flash memory is accessed as internal memory. In this mode, the configuration of OSPI needs to be completed before accessing the memory for the first time. The specific configuration is as follows:

1. Configure the read operation frame timing by setting OSPI_TIMCFG register.
2. Configure the read frame format by setting OSPI_TCFG register.
3. Specify the instruction to be sent to the external flash memory by set OSPI_INS register.
4. Specify the alternate bytes to be sent to the external flash memory immediately after the address is sent by setting OSPI_ALTE register.
5. Configure the write operation frame timing by setting OSPI_WTIMCFG register.
6. Configure the write frame format by setting OSPI_WTCFG register.
7. Specify the instruction to be sent to the external flash memory by set OSPI_WINS register.
8. Specify the alternate bytes to be sent to the external flash memory immediately after the address is sent by setting OSPI_WALTE register.

## 29.6. Data sampling shift

The OSPI samples data 1/2 of a clock after the data is driven by the external memory after reset. The data samples can be shifted 1/2 clock cycle by configuring the SSAMPLE bit in OSPI_TIMCFG register. Software must clear the SSAMPLE bit when DADTR is set.

## 29.7. Busy

BUSY bit is set once the OSPI start to operate the external flash memory.

In indirect mode, BUSY is reset if the command phase is end and the FIFO is empty. In status polling mode, only when a match occurs, the BUSY bit will be reset.

## 29.8. Error management

An error can be generated in the following case.

In indirect or status polling, TERR is generated immediately when a wrong address has been

programed in ADDR register according to MESZ[4:0].

In indirect mode, if the address (ADDR) plus data length (DTLEN) is greater than external memory size, TERR will be set once the OSPI is triggered.

In memory mapped mode, when an out of range access is done by AXI master will generate an AXI error.

## 29.9. OSPI interrupt

**Table 29-2 OSPI interrupt requests**

| Flag | Description | Clear method | Interrupt enable bit |
|---|---|---|---|
| FT | FIFO threshold | By hardware | FTIE |
| TC | Transfer complete | Set TCC bit in OSPI_STATC register | TCIE |
| TERR | Transfer error | Set TERRC bit in OSPI_STATC register | TERRIE |
| SM | Status match | Set SMC bit in OSPI_STATC register | SMIE |

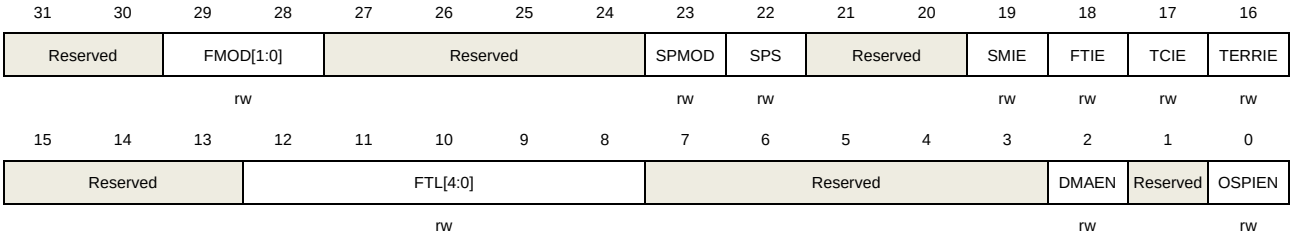
## 29.10. Register definition

OSPI0 base address: 0x5200 5000
OSPI1 base address: 0x5200 A000

### 29.10.1. Control register (OSPI_CTL)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by word(32-bit)
This register cannot be modified when the BUSY bit is 1.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:28 | FMOD[1:0] | Functional mode<br>00: Indirect write mode<br>01: Indirect read mode<br>10: Status polling mode<br>11: Memory mapped mode<br>If DMAEN = 1, then the DMA controller for the corresponding channel must be disabled before changing the FMOD value |
| 27:24 | Reserved | Must be kept at reset value. |
| 23 | SPMOD | Status polling match mode<br>0: AND match mode. SM is set if all the unmasked bits received from the flash memory match the corresponding bits in the match register.<br>1: OR match mode. SM is set if any one of the unmasked bits received from the flash memory matches its corresponding bit in the match register. |
| 22 | SPS | Status polling mode stop<br>This bit determines if status polling is stopped after a match.<br>0: Reserved.<br>1: Status polling mode stops as soon as there is a match. |
| 21:20 | Reserved | Must be kept at reset value. |
| 19 | SMIE | Status match interrupt enable<br>0: Disable status match interrupt. |

| | | |
|---|---|---|
| | | 1: Enable status match interrupt. |
| 18 | FTIE | FIFO threshold interrupt enable<br>0: Disable FIFO threshold interrupt.<br>1: Enable FIFO threshold interrupt. |
| 17 | TCIE | Transfer complete interrupt enable<br>0: Disabe transfer complete interrupt.<br>1: Enabe transfer complete interrupt. |
| 16 | TERRIE | Transfer error interrupt enable<br>0: Disable transfer error interrupt.<br>1: Enable transfer error interrupt. |
| 15:13 | Reserved | Must be kept at reset value. |
| 12:8 | FTL[4:0] | FIFO threshold level<br>This bit are useful in indirect mode, the threshold number of bytes in the FIFO that will cause the FIFO threshold flag to be set.<br>In indirect write mode (FMOD = 00):<br>0: FT is set if there are 1 or more free bytes available to be written to the FIFO.<br>1: FT is set if there are 2 or more free bytes available to be written to the FIFO.<br>...<br>31: FT is set if there are 32 free bytes available to be written to the FIFO.<br>In indirect read mode (FMOD = 01)：<br>0: FT is set if there are 1 or more free bytes available to be read from the FIFO.<br>1: FT is set if there are 2 or more free bytes available to be read from the FIFO.<br>...<br>31: FT is set if there are 32 free bytes available to be read from the FIFO.<br>If DMAEN = 1, then the DMA controller for the corresponding channel must be disabled before changing the FTL value. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | DMAEN | DMA enable<br>In indirect mode, DMA can be used to transfer data through OSPI_DATA register. DMA transfers are initiated when FT is set.<br>0: DMA disabled.<br>1: DMA enabled. |
| 1 | Reserved | Must be kept at reset value. |
| 0 | OSPIEN | Enable OSPI<br>0: Disable OSPI.<br>1: Enable OSPI. |

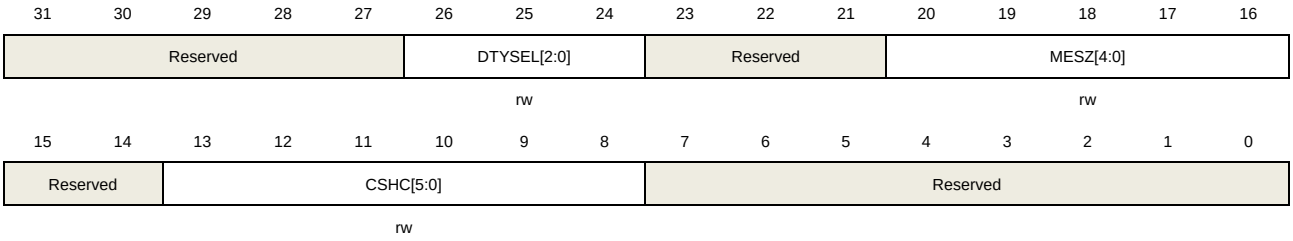
### 29.10.2. Device configuration register 0 (OSPI_DCFG0)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26:24 | DTYSEL[2:0] | Select device type<br>000: Micron mode, D0/D1 ordering in DTR 8 bit data mode. Regular SPI protocol in single/dual/quad/octal line mode.<br>001: Macronix mode, D1/D0 ordering in DTR 8 bit data mode. Regular SPI protocol in single/dual/quad/octal line mode.<br>010: Standard mode.<br>011: Macronix RAM mode, D1/D0 ordering in DTR 8 bit data mode. Regular SPI protocol in single/dual/quad/octal line mode with dedicated address mapping.<br>Others: Reserved. |
| 23:21 | Reserved | Must be kept at reset value. |
| 20:16 | MESZ[4:0] | Memory size<br>This field defines the size of external memory using the following formula:<br>Number of bytes in memory = $2^{[MESZ[4:0]+1]}$<br>MESZ[4:0]+1 is effectively the number of address bits in the memory. The memory capacity can be up to 4GB in indirect mode, while it is limited to 256MB in memory mapped mode. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:8 | CSHC[5:0] | Chip select high cycle<br>CSHC+1 dedines the minimum number of CLK cycle which the CSN must stay high between two command sequences.<br>0: CSN stays high for at least 1 cycle between memory commands.<br>1: CSN stays high for at least 2 cycles between memory commands.<br>...<br>63: CSN stays high for at least 64 cycles between memory commands. |
| 7:0 | Reserved | Must be kept at reset value. |

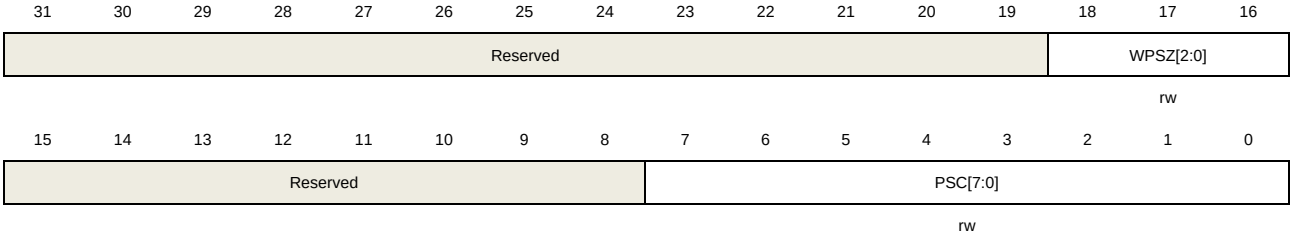
### 29.10.3. Device configuration register 1 (OSPI_DFCG1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by word(32-bit)

This register cannot be modified when the BUSY bit is 1.



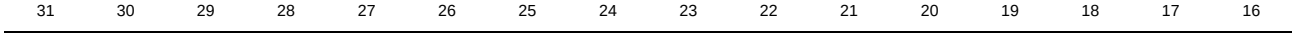

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18:16 | WPSZ[2:0] | Wrap size<br>000: The external memory does not support wrap read.<br>001: Reserved.<br>010: External memory device supports wrap size of 16 bytes.<br>011: External memory device supports wrap size of 32 bytes.<br>100: External memory device supports wrap size of 64 bytes.<br>101: External memory device supports wrap size of 128 bytes.<br>110: Reserved<br>111: Reserved |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | PSC[7:0] | This field defines the scaler factor for generating SCK based on the kernel clock (value+1)<br>0：$F_{CLK} = F_{KERNEL}$<br>1：$F_{CLK} = F_{KERNEL} / 2$<br>2：$F_{CLK} = F_{KERNEL} / 3$<br>…<br>255：$F_{CLK} = F_{KERNEL} / 256$<br>For odd clock division factors, CLK's duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high. |

### 29.10.4. Status register (OSPI_STAT)

Address offset: 0x20

Reset value: 0x0000 0004

This register can be accessed by word(32-bit).



| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FL[5:0] | | | | | | Reserved | | BUSY | Reserved | SM | FT | TC | TERR |
| | | r | | | | | | | | r | | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:8 | FL[5:0] | FIFO level<br>This field gives the number of valid bytes which are being held in the FIFO. In status polling mode, FL is zero. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | BUSY | Busy<br>This bit is set when a command is transferring. This bit is cleared once the operation with the flash memory is finished and the FIFO is empty. |
| 4 | Reserved | Must be kept at reset value. |
| 3 | SM | Status match flag<br>This bit is set in status polling mode when the unmasked received data matches the expected value. It is cleared by writing 1 to SMC. |
| 2 | FT | FIFO threshold flag<br>In indirect mode, this bit is set when the FIFO threshold has been reached, or if the FIFO is not empty after the last read operation from the flash memory.<br>In status polling mode, this bit is set when the status resgister is read from the external flash, and it is cleared when OSPI_DATA is read. |
| 1 | TC | Transfer complete flag<br>This bit is set in indirect mode when the programmed number of data has been transmitted. It is cleared by writing 1 TCC. |
| 0 | TERR | This bit is set when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to TERRC. |

### 29.10.5. Status clear register (OSPI_STATC)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | SMC | Reserved | TCC | TERRC |
|---|---|---|---|---|
| | w | | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | SMC | Clear status match flag<br>Writing 1 clears the SM flag in the OSPI_STAT register. |
| 2 | Reserved | Must be kept at reset value. |
| 1 | TCC | Clear transfer complete flag<br>Writing 1 clears the TC flag in the OSPI_STAT register. |
| 0 | TERRC | Clear transfer error flag<br>Writing 1 clears the TERR flag in the OSPI_STAT register. |

### 29.10.6. Data length register (OSPI_DTLEN)

Address offset: 0x40
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DTLEN[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DTLEN[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DTLEN[31:0] | Data length<br>The data length is DTLEN+1 in indirect and status poling mode. The value of DTLEN no greater than 3 should be used for status polling mode. All 1's in indirect mode means undefined length, where OSPI will continue until the end of memory, ad defined by MESZ[4:0].<br>0x0000 0000: 1 byte will be transferred<br>0x0000 0001: 2 bytes will be transferred<br>0x0000 0002: 3 bytes will be transferred<br>0x0000 0003: 4 bytes will be transferred<br>...<br>0xFFFF FFFD: 4,294,967,294 (4G-2) bytes will be transferred<br>0xFFFF FFFE: 4,294,967,295 (4G-1) bytes will be transferred<br>0xFFFF_FFFF: undefined length – all bytes until the end of flash memory (as defined by MESZ[4:0]) are to be transferred. Continue reading indefinitely if |

MESZ[4:0] = 0x1F.

This field has no effect when in memory-mapped mode.

## 29.10.7. Address register(OSPI_ADDR)

Address offset: 0x48

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR[31:0] | Address<br>Address to be send to the external flash memory.<br>This bit field can only be written when the BUSY bit is 0 and the memory mapping mode is not configured. |

## 29.10.8. Data register (OSPI_DATA)

Address offset: 0x50

Reset value: 0x0000 0000

This register can be accessed by word/half word/byte(32-bit/16-bit/8-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DATA[31:0] | Data<br>Data to be transferred through the flash memory<br>In indirect write mode, data written to this register is stored on the FIFO before sent to the flash memory. If the FIFO is full, a write operation is stalled until the FIFO has enough space.<br>In indirect read mode, reading this register gives the data received from the flash memory. If the FIFO does not have as many bytes as requested by the read command and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete. |

In status polling mode, this register contains the last data read from the flash memory.

## 29.10.9. Status mask register (OSPI_STATMK)

Address offset: 0x80
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MASK[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MASK [15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MASK[31:0] | Status mask<br>Mask to be applied to the status bytes received from the flash memory.<br>For bit n:<br>0: Bit n of the data received is masked and its value is not considered in the matching logic.<br>1: Bit n of the data received is unmasked and its considered in the matching logic. |

## 29.10.10. Status match register(OSPI_STATMATCH)

Address offset: 0x88
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MATCH[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MATCH[15:0] | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MATCH[31:0] | Status match<br>Expected value to be compared with the masked status register to get a match. |

## 29.10.11. Interval register (OSPI_INTERVAL)

Address offset: 0x90

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INTERVAL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | INTERVAL[15:0] | Interval cycle<br>Number of SCK cycles between two read commands in status polling mode. |

## 29.10.12. Transfer configuration register (OSPI_TCFG)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DADTR | DATAMOD[2:0] | | | Reserved | | ALTESZ[1:0] | | ABDTR | ALTEMOD[2:0] | | |
| | | | | rw | rw | | | | | rw | | rw | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ADDRSZ[1:0] | | ADDRDTR | ADDRMOD[2:0] | | | Reserved | | INSSZ[1:0] | | Reserved | IMOD[2:0] | | |
| | | rw | | rw | rw | | | | | rw | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27 | DADTR | Data double transfer rate<br>0: Disable DTR mode in data phase<br>1: Enable DTR mode in data phase<br><br>**Note:** only for GD25LX512ME. |
| 26:24 | DATAMOD[2:0] | Data mode<br>This bit defines the operation mode of the data phase.<br>000: No data.<br>001: Data on a single line. |

| | | |
|---|---|---|
| | | 010: Data on two lines.<br>011: Data on four lines.<br>100: Data on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 23:22 | Reserved | Must be kept at reset value. |
| 21:20 | ALTESZ[1:0] | Alternate bytes size<br>This bit defines alternate bytes size.<br>00: 8-bit alternate byte<br>01: 16-bit alternate bytes<br>10: 24-bit alternate bytes<br>11: 32-bit alternate bytes |
| 19 | ABDTR | Alternate bytes double transfer rate<br>0: Disable DTR mode in alternate bytes phase<br>1: Enable DTR mode in alternate bytes phase<br>**Note:** only for GD25LX512ME. |
| 18:16 | ALTEMOD[2:0] | Alternate bytes mode<br>This field defines the alternate bytes phase mode of operation.<br>000: No alternate bytes.<br>001: Alternate bytes on a single line.<br>010: Alternate bytes on two lines.<br>011: Alternate bytes on four lines.<br>100: Alternate bytes on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:12 | ADDRSZ[1:0] | Address size<br>This bit defines address size.<br>00: 8-bit address.<br>01: 16-bit address.<br>10: 24-bit address.<br>11: 32-bit address. |
| 11 | ADDRDTR | Address double transfer rate<br>0: Disable DTR mode in address phase<br>1: Enable DTR mode in address phase<br>**Note:** only for GD25LX512ME. |
| 10:8 | ADDRMOD[2:0] | Address mode<br>This field defines the address phase mode of operation. |

| | | |
|---|---|---|
| | | 000: No address.<br>001: Address on a single line.<br>010: Address on two lines.<br>011: Address on four lines.<br>100: Address on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | INSSZ[1:0] | Instruction size<br>This field defines instruction size.<br>00: 8-bit instruction.<br>01: 16-bit instruction.<br>10: 24-bit instruction.<br>11: 32-bit instruction. |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | IMOD[2:0] | Instruction mode<br>This field defines the instruction phase mode of operation.<br>000: No instruction.<br>001: Instruction on a single line.<br>010: Instruction on two lines.<br>011: Instruction on four lines.<br>100: Instruction on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |

## 29.10.13. Timing configuration register (OSPI_TIMCFG)

Address: 0x108

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SSAMPLE | Reserved | | DEHQC | Reserved | | | | | | | | | | |
| | rw | | | rw | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | DUMYC[4:0] | | | | |
| | | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | SSAMPLE | Sample shift<br>By default, the OSPI samples data 1/2 of a SCK cycle after the data is driven by the flash memory. This bit allows the data is to be sampled later in order to account for exrernal signal delays.<br>0: No shift.<br>1: 1/2 cycle shift.<br>**Note:** the SSAMPLE bit must be set to 1 when communication rate greater than 40M. |
| 29:28 | Reserved | Must be kept at reset value. |
| 27 | DEHQC | Delay hold 1/4 cycle<br>0: No delay hold.<br>1: delay hold 1/4 cycle. |
| 26:5 | Reserved | Must be kept at reset value. |
| 4:0 | DUMYC[4:0] | Number of dummy cycles<br>This bit field defines the duration of the dummy instruction phase. |

## 29.10.14. Instruction register (OSPI_INS)

Address: 0x110
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSTRUCTION[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSTRUCTION[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | INSTRUCTION[31:0] | Instruction<br>Instruction to be send to the external flash memory. |

## 29.10.15. Alternate bytes register (OSPI_ALTE)

Address offset: 0x120
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ALTE[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ALTE[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ALTE[31:0] | Alternate bytes<br>Alternate bytes to be send to the external flash memory. |

## 29.10.16. Wrap transfer configuration register (OSPI_WPTCFG)

Address offset: 0x140

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DADTR | DATAMOD[2:0] | | | Reserved | | ALTESZ[1:0] | | ABDTR | ALTEMOD[1:0] | | |
| | | | | rw | rw | | | | | rw | | rw | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ADDRSZ[1:0] | | ADDRDTR | ADDRMOD[2:0] | | | Reserved | | INSSZ[1:0] | | Reserved | IMOD[2:0] | | |
| | | rw | | rw | rw | | | | | rw | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27 | DADTR | Data double transfer rate<br>0: Disable DTR mode in data phase.<br>1: Enable DTR mode in data phase.<br>**Note:** only for GD25LX512ME. |
| 26:24 | DATAMOD[2:0] | Data mode<br>This bit defines the operation mode of the data phase.<br>000: No data.<br>001: Data on a single line.<br>010: Data on two lines.<br>011: Data on four lines.<br>100: Data on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 23:22 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 21:20 | ALTESZ[1:0] | Alternate bytes size<br>This bit defines alternate bytes size.<br>00: 8-bit alternate byte<br>01: 16-bit alternate bytes<br>10: 24-bit alternate bytes<br>11: 32-bit alternate bytes |
| 19 | ABDTR | Alternate bytes double transfer rate<br>0: Disable DTR mode in alternate bytes phase<br>1: Enable DTR mode in alternate bytes phase<br>**Note:** only for GD25LX512ME. |
| 18:16 | ALTEMOD[2:0] | Alternate bytes mode<br>This field defines the alternate bytes phase mode of operation.<br>000: No alternate bytes.<br>001: Alternate bytes on a single line.<br>010: Alternate bytes on two lines.<br>011: Alternate bytes on four lines.<br>100: Alternate bytes on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:12 | ADDRSZ[1:0] | Address size<br>This bit defines address size.<br>00: 8-bit address.<br>01: 16-bit address.<br>10: 24-bit address.<br>11: 32-bit address. |
| 11 | ADDRDTR | Address double transfer rate<br>0: Disable DTR mode in address phase<br>1: Enable DTR mode in address phase<br>**Note:** only for GD25LX512ME. |
| 10:8 | ADDRMOD[2:0] | Address mode<br>This field defines the address phase mode of operation.<br>000: No address.<br>001: Address on a single line.<br>010: Address on two lines.<br>011: Address on four lines.<br>100: Address on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |

| | | |
|---|---|---|
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | INSSZ[1:0] | Instruction size<br>This field defines instruction size.<br>00: 8-bit instruction.<br>01: 16-bit instruction.<br>10: 24-bit instruction.<br>11: 32-bit instruction. |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | IMOD[2:0] | Instruction mode<br>This field defines the instruction phase mode of operation.<br>000: No instruction.<br>001: Instruction on a single line.<br>010: Instruction on two lines.<br>011: Instruction on four lines.<br>100: Instruction on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |

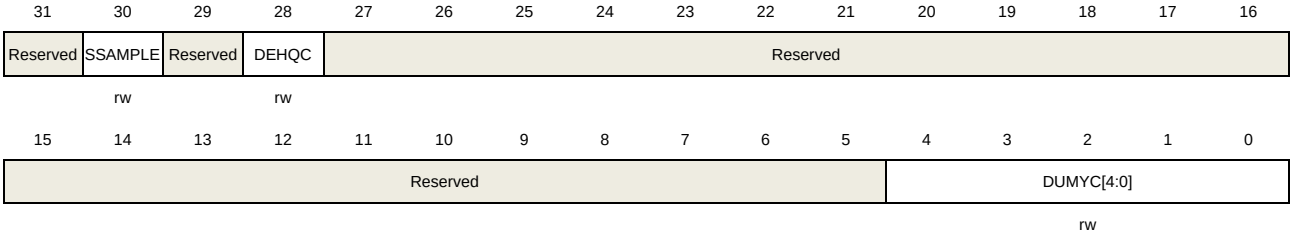
## 29.10.17. Wrap timing configuration register (OSPI_WPTIMCFG)

Address offset: 0x148

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | SSAMPLE | Sample shift<br>By default, the OSPI samples data 1/2 of a SCK cycle after the data is driven by the flash memory. This bit allows the data is to be sampled later in order to account for exrernal signal delays.<br>0: No shift.<br>1: 1/2 cycle shift. |

| | | |
|---|---|---|
| 29 | Reserved | Must be kept at reset value. |
| 28 | DEHQC | Delay hold 1/4 cycle<br>0: No delay hold.<br>1: delay hold 1/4 cycle. |
| 27:5 | Reserved | Must be kept at reset value. |
| 4:0 | DUMYC[4:0] | Number of dummy cycles<br>This bit field defines the duration of the dummy instruction phase. |

### 29.10.18. Wrap instruction register (OSPI_WPINS)

Address offset: 0x150
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSTRUCTION[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INSTRUCTION[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | INSTRUCTION[31:0] | Instruction<br>Instruction to be send to the external flash memory. |

### 29.10.19. Wrap alternate byte register (OSPI_WPALTE)

Address offset: 0x160
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).
This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ALTE[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ALTE[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ALTE[31:0] | Alternate bytes |

Alternate bytes to be send to the external flash memory.

## 29.10.20. Write transfer configuration register (OSPI_WTCFG)

Address offset: 0x180

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DADTR | DATAMOD[2:0] | | | Reserved | | ALTESZ[1:0] | | ABDTR | ALTEMOD[1:0] | | |
| | | | | rw | rw | | | | | rw | | rw | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ADDRSZ[1:0] | | ADDRDTR | ADDRMOD[2:0] | | | Reserved | | INSSZ[1:0] | | Reserved | IMOD[2:0] | | |
| | | rw | | rw | rw | | | | | rw | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27 | DADTR | Data double transfer rate<br>0: Disable DTR mode in data phase.<br>1: Enable DTR mode in data phase.<br>**Note:** only for GD25LX512ME. |
| 26:24 | DATAMOD[2:0] | Data mode<br>This bit defines the operation mode of the data phase.<br>000: No data.<br>001: Data on a single line.<br>010: Data on two lines.<br>011: Data on four lines.<br>100: Data on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 23:22 | Reserved | Must be kept at reset value. |
| 21:20 | ALTESZ[1:0] | Alternate bytes size<br>This bit defines alternate bytes size.<br>00: 8-bit alternate byte<br>01: 16-bit alternate bytes<br>10: 24-bit alternate bytes<br>11: 32-bit alternate bytes |
| 19 | ABDTR | Alternate bytes double transfer rate<br>0: Disable DTR mode in alternate bytes phase<br>1: Enable DTR mode in alternate bytes phase |

| | | |
|---|---|---|
| | | **Note:** only for GD25LX512ME. |
| 18:16 | ALTEMOD[2:0] | Alternate bytes mode<br>This field defines the alternate bytes phase mode of operation.<br>000: No alternate bytes.<br>001: Alternate bytes on a single line.<br>010: Alternate bytes on two lines.<br>011: Alternate bytes on four lines.<br>100: Alternate bytes on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:12 | ADDRSZ[1:0] | Address size<br>This bit defines address size.<br>00: 8-bit address.<br>01: 16-bit address.<br>10: 24-bit address.<br>11: 32-bit address. |
| 11 | ADDRDTR | Address double transfer rate<br>0: Disable DTR mode in address phase.<br>1: Enable DTR mode in address phase.<br>**Note:** only for GD25LX512ME. |
| 10:8 | ADDRMOD[2:0] | Address mode<br>This field defines the address phase mode of operation.<br>000: No address.<br>001: Address on a single line.<br>010: Address on two lines.<br>011: Address on four lines.<br>100: Address on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | INSSZ[1:0] | Instruction size<br>This field defines instruction size.<br>00: 8-bit instruction.<br>01: 16-bit instruction.<br>10: 24-bit instruction.<br>11: 32-bit instruction. |
| 3 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 2:0 | IMOD[2:0] | Instruction mode<br>This field defines the instruction phase mode of operation.<br>000: No instruction.<br>001: Instruction on a single line.<br>010: Instruction on two lines.<br>011: Instruction on four lines.<br>100: Instruction on eight lines.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved. |

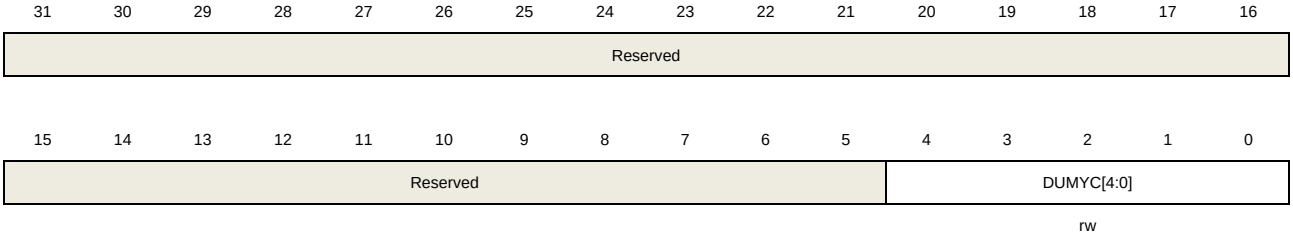
## 29.10.21. Write timing configuration register (OSPI_WTIMCFG)

Address offset: 0x188

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4:0 | DUMYC[4:0] | Number of dummy cycles<br>This bit field defines the duration of the dummy instruction phase. |

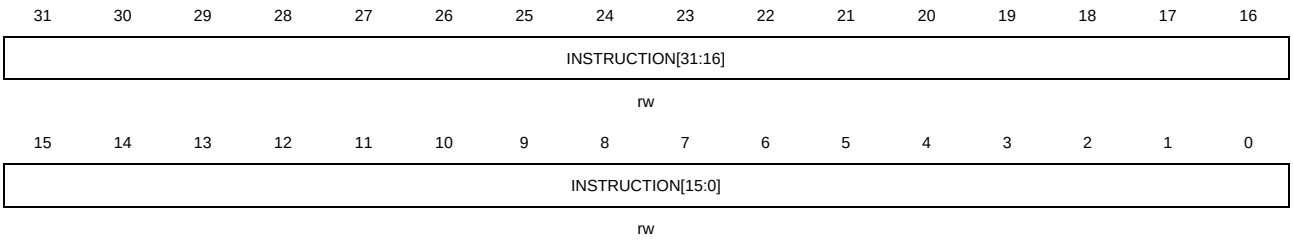
## 29.10.22. Write instruction register (OSPI_WINS)

Address offset: 0x190

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.



| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | INSTRUCTION[31:0] | Instruction<br>Instruction to be send to the external flash memory. |

### 29.10.23. Write alternate byte register (OSPI_WALTE)

Address offset: 0x1A0

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

This register cannot be modified when the BUSY bit is 1.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ALTE[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ALTE[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ALTE[31:0] | Alternate bytes<br>Alternate bytes to be send to the external flash memory. |

# 30. Clock phase delay module (CPDM) (For GD32H737_757_759)

## 30.1. Overview

The Clock Phase Delay Module (CPDM) is used to delay the phase of the input clock and then output the clock. When used, the application needs to first program the phase of the output clock, and then use the output clock in other peripherals to receive data.

Phase delay is related to voltage and temperature and may require reconfiguration of the application and redetermination of the phase relationship between the output clock and the received data as parameters change.

## 30.2. Characteristics

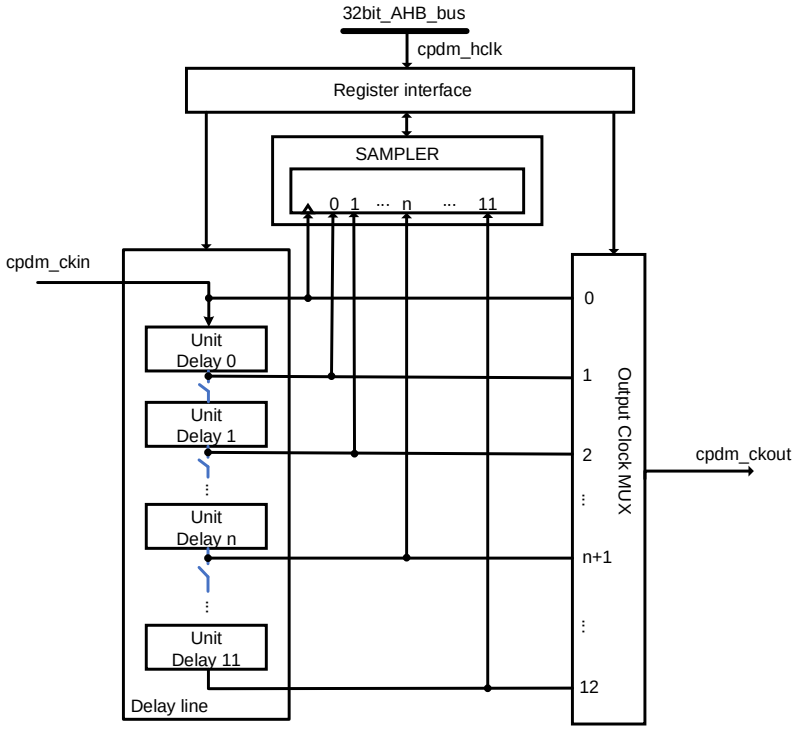
- Supports the input clock frequency ranges: 25 MHz ~ 208MHz.
- Supports up to 12 output clock phase selections.

## 30.3. Function overview

The Clock Phase Delay Module (CPDM) consists of register interface module, delay line module, delay line length sampling module and output clock selection multiplexer. The delay line module is used to generate unit delay.

**Note:**
CPDM is only used in SDIO, please refer to ***Figure 36-6. SDIO block diagram***.

**Figure 30-1. Schematic diagram of CPDM**




**Note:**

cpdm_hclk: register interface clock of CPDM.
cpdm_ckin: the input clock of CPDM.
cpdm_ckout: the output clock of CPDM.

## 30.3.1. Overview

The Clock Phase Delay Module (CPDM) can be enabled by configuring the CPDMEN bit in the CPDM_CTL register. The delay line length sampling module can be enabled by configuring the DLSEN bit in the CPDM_CTL register.

Before shifting the phase of the input clock, the delay line length should be configured as one input clock period. When the CPDM is enabled and the delay line length sampling module is enabled (DLSEN = 1), define the number of delay steps required for a unit delay unit by configuring the DLSTCNT[6:0] bit field in the CPDM_CFG register. When the delay line length sampling module is enabled (DLSEN = 1), the length sampler can access the delay line length (DLLEN) and the length valid flag (DLLENF) in the CPDM_CFG register. Therefore, the delay line length can be configured by enabling the length sampler. When the delay line length sampling module is disabled (DLSEN = 0), the CPSEL bit in the CPDM_CFG register is used to select the output clock phase.

When the delay line length configuration is complete and the delay line length sampling module is disabled (DLSEN = 0), the clock that passes through the phase shift can be finally output through the clock phase selector.

- When CPDMEN = 0 and DLSEN = 0, CPDM output clock is enabled, and the output clock is equal to the input clock.
- When DLSEN = 1, the delay sampling module is enabled, and the CPDM output clock is disabled.
- When CPDMEN = 1 and DLSEN = 0, the CPDM output clock is enabled, and the phase of the output clock is determined by the configuration of CPSEL[3:0] bits in CPDM_CFG.

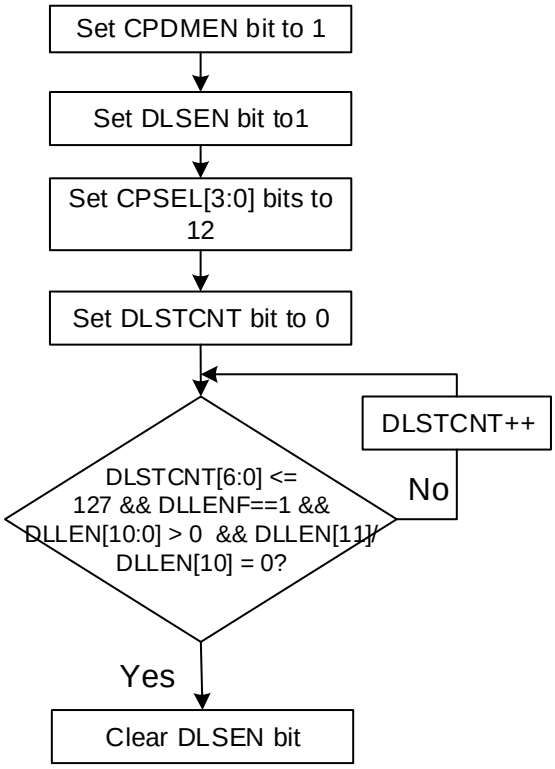
**Note:** The unit delay and output clock phase can only be changed when DLSEN is 1.

## 30.3.2. Operation process

### Step one

The delay line length needs to be configured before configuring the output clock phase. The configured delay line length should cover a complete input clock cycle. ***Figure 30-2. CPDM delay line length configuration flowchart*** shows the configuration process.

**Figure 30-2. CPDM delay line length configuration flowchart**



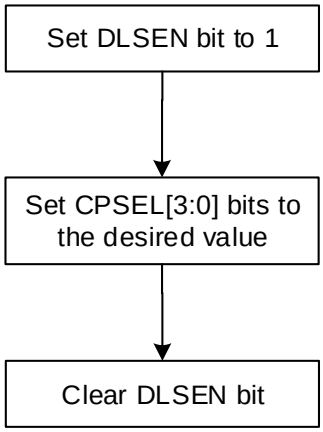

**Note:**

1. If DLLEN[10:0] > 0 and DLLEN[11] / DLLEN[10] = 0, configure the delay line length to an input clock cycle.
2. For N = 10 to 0, if DLLEN[N] = 1, the number of unit delays covering an input clock cycle is N.

**Step two**

The user can configure any output clock phase between unit delay N when the delay line length has been configured for one full input clock cycle. The specific method is as follows *Figure 30-3. CPDM output clock phase configuration flowchart* .

**Figure 30-3. CPDM output clock phase configuration flowchart**



## 30.4. Register definition

CPDM(SDIO0) base address: 0x5200 8000

CPDM(SDIO1) base address: 0x4802 2800

### 30.4.1. Control register (CPDM_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | DLSEN | CPDMEN |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | DLSEN | CPDM delay line sample module enable bit<br>0: disable CPDM delay line sample module<br>1: enable CPDM delay line sample module |
| 0 | CPDMEN | CPDM enable bit<br>0: disable CPDM<br>1: enable CPDM |

### 30.4.2. Configuration register (CPDM_CFG)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DLLENF | Reserved | | | DLLEN[11:0] | | | | | | | | | | | |
| r | | | | r | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | DLSTCNT[6:0] | | | | | | | Reserved | | | | CPSEL[3:0] | | | |
| | rw | | | | | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DLLENF | Valid mark of delay line length<br>0: The length value in DLLEN[11:0] is invalid. |

| | | |
|---|---|---|
| | | 1: The length value in DLLEN[11:0] is valid. |
| 30:28 | Reserved | Must be kept at reset value. |
| 27:16 | DLLEN[11:0] | Delay line length<br>12 unit delay values sampled on the rising edge of the input clock<br>Only valid if DLLENF = 1. |
| 15 | Reserved | Must be kept at reset value. |
| 14:8 | DLSTCNT[6:0] | Defines a delay step count for a unit delay UNIT<br>This bits can be written only when DLSEN = 1.<br>0000000: UNIT delay = Initial delay<br>0000001: UNIT delay = Initial delay + 1 * Delay step<br>…<br>1111111: UNIT delay = Initial delay + 127 * Delay step |
| 7:4 | Reserved | Must be kept at reset value. |
| 3:0 | CPSEL[3:0] | Output clock phase selection<br>These bits can be written only when DLSEN = 1.<br>0000: Output clock phase = input clock<br>0001: Output clock phase = input clock + 1 * UNIT delay<br>..<br>1100: Output clock phase = input clock + 12 * UNIT delay<br>1101 ~ 1111: Reserved |

# 31. Digital camera interface (DCI) (For GD32H737_757_759)

## 31.1. Overview

DCI is a parallel interface to capture video or picture from a camera. It supports various color space such as YUV/RGB, as well as compression format such as JPEG. Support CCIR656 video decoder formats and perform additional processing of the image.

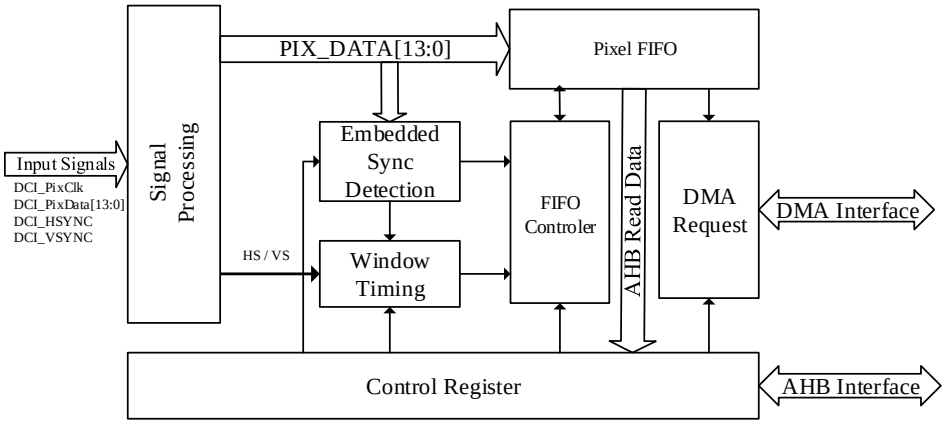
## 31.2. Characteristics

- Digital video/picture capture
- 8/10/12/14 data width supported
- High transfer efficiency with DMA interface
- Video/picture crop supported
- Various pixel digital encoding formats supported including YCbCr422/RGB565/YUV420/Bayer
- JPEG compression format supported
- Hard/embedded synchronous signals supported
- Support for CCIR656 video interface as well as traditional sensor interface.

## 31.3. Block diagram

The DCI contains these modules: Signal Processing, Pixel FIFO, FIFO controller, window timing, embedded sync detection, DMA interface and control register.

**Figure 31-1. DCI module block diagram**




The signal processing module generates useful signals for other internal modules from external input signals. The frequency of HCLK should be 2.5 times higher than DCI_PixClk to

ensure the proper operation of signal processing module.

The embedded sync detection module is designed to support embedded synchronization mode. In DCI embedded synchronization mode, video synchronization information is embedded into pixel data and there is no hardware horizontal or vertical synchronization signal (DCI_HSYNC or DCI_VSYNC). DCI uses embedded sync detection module to extract synchronization information from pixel data, and then recover horizontal and vertical synchronization signals.

The window timing module performs image cutting function. This module calculates a pixel's position using synchronization signals either from DCI interface or embedded sync detection module and then decides whether this pixel data needs to be received according to the configuration of DCI_CWSPOS and DCI_CWSZ registers.

DCI uses a 4 word (32-bit) FIFO to buffer the received pixel data. If DMA mode is enabled, the DMA interface asserts a DMA request every time a 32-bit data is received. Control register provides register interface between DCI and software.

## 31.4. Signal description

**Table 31-1. PINs used by DCI**

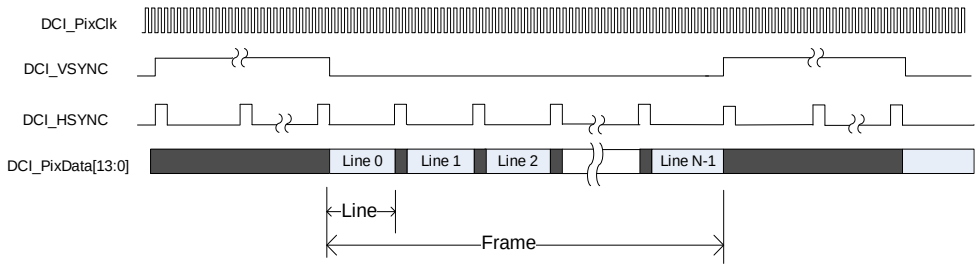
| Direction | Pins | Signals | Width | Description |
|---|---|---|---|---|
| Input | DCI_PIXCLK | DCI_PixClk | 1 | DCI Pixel Clock |
| Input | DCI_D | DCI_PixData | 14 | DCI Pixel Data |
| Input | DCI_HSYNC | DCI_HSYNC | 1 | DCI Horizontal Synchronization |
| Input | DCI_VSYNC | DCI_VSYNC | 1 | DCI Vertical Synchronization |

## 31.5. Function overview

### 31.5.1. DCI hardware synchronization mode

In DCI hardware synchronization mode (ESM bit in DCI_CTL register is 0), DCI_HSYNC and DCI_VSYNC signals are used to indicate the start of a line and a frame. DCI captures pixel data from DCI_PixData[13:0] at rising or falling edge of DCI_PixClk (clock polarity is configured by CKS bit in DCI_CTL).

**Figure 31-2. Hardware synchronization mode**

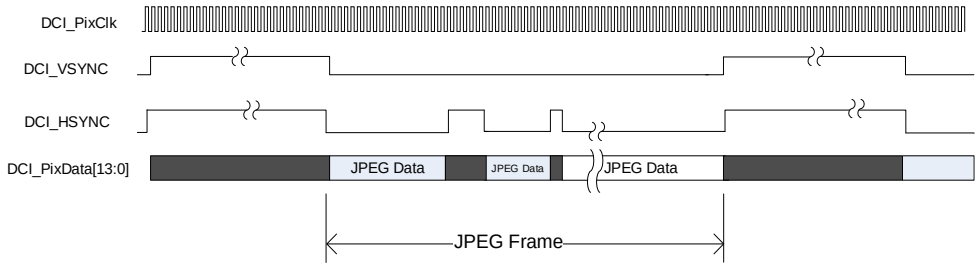


The above figure assumes that the polarities of both DCI_HSYNC and DCI_VSYNC are high during blanking period, so the data on DCI_PixData lines is only valid when both DCI_HSYNC and DCI_VSYNC are low.

#### JPEG mode

DCI supports JPEG video/picture compression format in hardware synchronization mode. In JPEG mode (JM bit in DCI_CTL is set), the DCI_VSYNC is used to indicate start of a new frame, and DCI_HSYNC is used as stream data valid signal.

**Figure 31-3. Hardware synchronization mode: JPEG format supporting**




### 31.5.2. Embedded synchronization mode

DCI supports embedded synchronization mode. In this mode there are only DCI_PixData and DCI_PixClk signals in DCI interface and the synchronization information is embedded in the pixel data. This mode is enabled by setting ESM bit and clearing JM bit in DCI_CTL register.

In embedded synchronization mode, line and frame synchronization information is encoded into sync code and embedded into the pixel data. There are four kinds of sync code: Line Start(LS), Line End(LE), Frame Start(FS) and Frame End(FE). In this mode the data width is forced to 8 and each sync code is composed by 4-byte sequence: FF-00-00-XY and XY is defined in DCI_SC register. In embedded synchronization mode, the 0xFF and 0x00 should not appear in pixel data to avoid mistake.

In embedded synchronization mode, DCI starts to detect the sync codes after enabled and recover line/frame synchronization information. For example, DCI starts to capture a new frame if it detects a Frame End code and then a Frame Start Code.

When detecting sync code, it is possible to make DCI compare only a few bits of XY byte in FF_00_00_XY sequence by configuring sync code unmask register (DCI_SCUMSK). DCI will only compare bits unmasked by DCI_SCUMSK register. For example: LS in DCI_SC register is A5 and LSM in DCI_SCUMSK is F0, then DCI will only compare the higher 4 bits for LS sync code and thus, FF-00-00-A6 sequence will also be detected as a LS code.

### 31.5.3. CCIR656 mode

#### Interlace mode

CCIR656 standard use an embedded timing codec to replace the VSYNC and HSYNC signal.

Only the PIXCLK and DCI_PixData[7:0] signals are used in interlace mode. Code (SAV) represents the beginning of each valid row, and code (EAV) is the end of each valid row.Blanking data inserts are inserted between the EAV and SAV code. The DCI decode the embedded timing codec, according to the information of the field, the original YCbCr image is rearranged in the software.

**Table 31-2. Typical one-line data composition**

| EAV Code | | | | Blanking Video | | | | SAV Code | | | | Active Video | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FF | 0 | 0 | XY | Cb | Y | Cr | V | FF | 0 | 0 | XY | Cb | Y | Cr | Y |
| 4 Bytes | | | | 280(268) Bytes | | | | 4 Bytes | | | | 1440 Bytes | | | |

The 4-byte format of the EAV and SAV is specified as follows (represented in hexadecimal below):FF 00 00 XY

The first three bytes are fixed and must be FF 00 00, while the fourth byte (XY) is determined by the field and the hidden information, and its eight bits are defined as follows: 1 F V H P3 P2 P1 P0.

F: marks the field information, which is 0 when transmitting the even field and 1 when transmitting the odd field.

V: Marks the blanking information, which is 1 when the blanking data is transmitted and 0 when the valid video data is transmitted.

H: Label EAV or SAV, where SAV is 0 and EAV is 1.

P0~P3 is the protection bit, and its value depends on F, H and V, which plays the role of calibration. The calculation method is as follows:

**Table 31-3. Coding for SAV and EAV**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| **1** | **F** | **V** | **H** | **P3** | **P2** | **P1** | **P0** |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

The DCI decodes and filters out the timing-coding from the data stream, recovering VSYNC and HSYNC signals for internal use, such as statistical block control. Data is forwarded sequentially to the data receiving without reordering, field 1 followed by field 0. The fields must be reordered in software to get back the original image.

The Change of Field interrupt Flag(COFIF) flag bit will be set when an alternation occurs on the spot. CCIR656 standard images are generally in 625/50 PAL or 525/60 NTSC format. The image is composed of odd and even fields, horizontal and vertical blanking and valid data. A

field consists of three parts, top vertical blanking, valid data and bottom vertical blanking.

The following table showss the frame information in PAL/NTSC system.

**Table 31-4. A frame image format definition**

| Lines | | Field/VBlk | | Line Description |
|---|---|---|---|---|
| PAL | NTSC | F | V | |
| 22 | 19 | 0 | 1 | Field 0 top vertical blanking |
| 288 | 240 | 0 | 0 | Field 0 active video |
| 2 | 3 | 0 | 1 | Field 0 bottom vertical blanking |
| 23 | 20 | 1 | 1 | Field 1 top vertical blanking |
| 288 | 240 | 1 | 0 | Field 1 active video |
| 2 | 3 | 1 | 1 | Field 1 bottom vertical blanking |
| 625 | 525 | | | |

### Progressive Mode

Images are scanned and arranged in a progressive manner.This activity field is considered to be field 1 and the DCI decoding ignores the F bit in the SAV/EAV code.

**Table 31-5. General case of progressive mode**

| | | | | |
|---|---|---|---|---|
| EAV | Blanking | SAV | Blanking | Field 1 (F = 1) |
| ⋮ | | | | |
| EAV | Blanking | SAV | Blanking | |
| EAV | Blanking | SAV | Data | |
| ⋮ | | | | |
| EAV | Blanking | SAV | Data | |

In progressive mode, COFIF will be ignored, but VSIF can generate an interrupt. When the VSYNC signal is provided by the camera sensor, it is called external VSYNC mode, and when the VSIF flag bit is decoded from the embedded code, it is called internal VSYNC mode. DCI can run both internal and external VSYNC mode.

### Error Correction for CCIR656 Coding

According to the CCIR coding algorithm, the guard bits in SAV and EAV are coded in a way that allows 1-bit errors to be corrected, or 2-bit errors to be detected by the decoder. This feature is supported by the interlaced mode CCIR decoder in DCI.

For 1-bit error conditions, the user can choose to correct the error automatically, or simply display it as a status flag. For 2-bit error cases, the error will only be displayed as a status flag since the decoder cannot make corrections.

When the CCIR error interrupt is enabled (CCEIE is set), an interrupt is generated when an error is detected. If automatic error correction is enabled (AECEN is set in the DCI_CTL register), 1-bit errors will be corrected automatically. If 2-bit errors are enabled, the CCEIF error flag will be set.

### 31.5.4. Capture data using snapshot or continuous capture modes

The DCI supports two capture modes: snapshot and continuous capture. Capture mode is configured by SNAP bit in DCI_CTL register.

After correctly configure, enable DCI and set CAP bit in DCI_CTL register, the DCI begins to detect frame start. It begins to capture data once a frame start is detected. In snapshot mode(SNAP=1), DCI automatically stops capturing and clears the CAP bit after a whole frame is captured completely, while in continuous mode, DCI prepares to capture the next frame. The DCI capture frequency is defined by FR[1:0] bits in continuous mode. For example, if FR[1:0]=00, DCI captures each frame, and if FR[1:0]=01, DCI only captures every alternate frame.

In continuous mode, software may clear the CAP bit any time when DCI is capturing data, but DCI doesn't stop capture immediately. It always stops after a complete frame ends. Software should read back the CAP bit to know whether the DCI stops effectively.

Use the end of the frame flag to count the frame, take part of the frame data.In CCIR656 interlaced scan mode, the end of Field1(the second field of the image,F=1) is considered to be the end of a frame.

### 31.5.5. Window function

DCI supports window function which is able to cut a part of image from the captured frame. Window function is enabled by setting WDEN bit in DCI_CTL register and this function is disabled in JPEG mode.

DCI continuously counts and calculates pixels' horizontal and vertical position during capturing, and compares the position and the values in crop window registers (DCI_CWSPOS and DCI_CWSZ), and then discards those pixels outside the crop window and only pushes pixels inside the window into the pixel FIFO.

If a frame ends when the vertical lines size defined in DCI_CWSZ is not reached yet, the end of frame flag will be triggered and DCI stops the capture. In CCIR656 interlaced scan mode, each field image is clipped.

### 31.5.6. Pixel formats, data padding and DMA

DCI supports various pixel digital encoding formats including YCbCr422/RGB565. However, DCI only receives these pixel data, pads these pixels into a word and push into a pixel FIFO. DCI doesn't perform any pixel format conversion or data processing and doesn't care about the detail of pixel format.

DCI uses a 32-bits width data buffer to transfer between DCI interface and pixel FIFO. These are two padding method in this module: byte padding and half-word padding, depending on the data width of DCI interface. Data width is configured by DCIF[1:0] in DCI_CTL register. The data width is fixed to 8 in JPEG mode and embedded synchronization mode.

The DMA interface sends DMA request when a 32-bit data is received.

### Byte padding mode

Byte padding mode is used if data width of DCI interface is 8. In byte padding mode four bytes are filled into the 32-bits width data buffer. In Non-JPEG mode, the DCI pushes the 32-bits buffer's data into the pixel FIFO when the buffer is full or meets the end of a line. In JPEG mode, the DCI pushes the 32-bits buffer's data into the pixel FIFO when the buffer is full or meets the end of a frame.

**Table 31-6. Memory view in byte padding mode**

| | | | |
|---|---|---|---|
| D3[7:0] | D2[7:0] | D1[7:0] | D0[7:0] |
| D7[7:0] | D6[7:0] | D5[7:0] | D4[7:0] |

### Half-word padding mode

Half-word padding is used if data width of DCI interface is configured into 10/12/14. In this mode each pixel data is extended into 16-bits length by filling zero at higher position, so the 32-bits width data buffer is able to hold two pixel data. DCI pushes the data buffer into pixel FIFO each time the buffer is full or line end.

**Table 31-7. Memory view in half-word padding mode**

| | | | |
|---|---|---|---|
| 2'b00 | D1[13:0] | 2'b00 | D0[13:0] |
| 2'b00 | D3[13:0] | 2'b00 | D2[13:0] |
| 2'b00 | D5[13:0] | 2'b00 | D4[13:0] |
| 2'b00 | D7[13:0] | 2'b00 | D6[13:0] |

## 31.6. Interrupts

There are several status and error flags in DCI, and interrupts may be asserted from these flags. These status and error flags will assert global DCI interrupt if enabled by corresponding bit in DCI_INTEN. These flags are cleared by writing 1 into DCI_INTC register.

**Table 31-8. Status/Error flags**

| Status Flag Name | Description |
|---|---|
| CCEIF | CCIR Error Interrupt Flag |
| COFIF | CCIR Change of Field Interrupt Flag |
| F1IF | CCIR Field 1 Interrupt Flag |
| F0IF | CCIR Field 0 Interrupt Flag |
| ELF | End of Line Flag |
| EFF | End of Frame Flag |
| OVRF | FIFO Overrun Flag |
| VSF | Frame VS Blank Flag |
| ESEF | Embedded Sync Error Flag |

## 31.7. Register definition

DCI base address: 0x4802 0000

### 31.7.1. Control register (DCI_CTL)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | EVSEN |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AECEN | DCIEN | CCMOD | CCEN | DCIF[1:0] | | FR[1:0] | | VPS | HPS | CKS | ESM | JM | WDEN | SNAP | CAP |
| rw | rw | rw | rw | rw | | rw | | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | EVSEN | External vsync enable<br>0: External vsync mode is disabled<br>1: External vsync mode is enabled<br>Only works in CCIR progressive mode |
| 15 | AECEN | Automatic error correction enable, 1bit correction<br>0: Automatic error correction is disabled<br>1: Automatic error correction is enabled<br>Only works in CCIR interlaced mode |
| 14 | DCIEN | DCI Enable<br>0: DCI is disabled<br>1: DCI is enabled |
| 13 | CCMOD | CCIR mode select<br>0: CCIR progressive mode<br>1: CCIR interlace mode |
| 12 | CCEN | CCIR enable<br>0: CCIR is disabled<br>1: CCIR is enabled |
| 11:10 | DCIF[1:0] | Digital camera interface format<br>00: 8-bit data on every pixel clock<br>01: 10-bit data on every pixel clock<br>10: 12-bit data on every pixel clock |

| | | |
|---|---|---|
| | | 11: 14-bit data on every pixel clock |
| 9:8 | FR[1:0] | Frame rate<br>FR defines the frame capture rate in continuous capture mode<br>00: Capture all frames<br>01: Capture one in 2 frames<br>10: Capture one in 4 frames<br>11: Reserved |
| 7 | VPS | Vertical polarity selection<br>0: Low level during blanking period<br>1: High level during blanking period |
| 6 | HPS | Horizontal polarity selection<br>0: Low level during blanking period<br>1: High level during blanking period |
| 5 | CKS | Clock polarity selection<br>0: Capture at falling edge<br>1: Capture at rising edge |
| 4 | ESM | Embedded synchronous mode<br>0: Embedded synchronous mode is disabled<br>1: Embedded synchronous mode is enabled |
| 3 | JM | JPEG mode<br>0: JPEG mode is disabled<br>1: JPEG mode is enabled |
| 2 | WDEN | Window enable<br>0: Window is disabled<br>1: Window is enabled |
| 1 | SNAP | Snapshot mode<br>0: Continuous capture mode<br>1: Snapshot capture mode |
| 0 | CAP | Capture enable<br>0: Frame not captured<br>1: Frame is captured |

### 31.7.2. Status register0 (DCI_STAT0)

Address offset: 0x04

Reset value: 0x0000 0003

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | FV | VS | HS |
| | | | | | | | | | | | | | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | FV | FIFO valid<br>0: No valid pixel data in FIFO<br>1: Valid pixel data in FIFO |
| 1 | VS | VS line status<br>0: Not in vertical blanking period<br>1: In vertical blanking period |
| 0 | HS | HS line status<br>0: Not in horizontal blanking period<br>1: In horizontal blanking period |

### 31.7.3. Status register1 (DCI_STAT1)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CCEF | COFF | F1F | F0F | ELF | VSF | ESEF | OVRF | EFF |
| | | | | | | | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | CCEF | CCIR error flag<br>0: No error occurs<br>1: Error is found on the SVA or EVA codes<br>Only works in CCIR interlaced mode |
| 7 | COFF | CCIR change of field flag<br>0: No change of field event occurs<br>1: Change of field event occurs<br>Only works in CCIR interlaced mode |
| 6 | F1F | CCIR field 1 |

| | | |
|---|---|---|
| | | 0: Current field is not 1<br>1: Current field is 1<br>Only works in CCIR interlaced mode(cleared automatically when current field does not match) |
| 5 | F0F | CCIR field 0<br>0: Current field is not 0<br>1: Current field is 0<br>Only works in CCIR interlaced mode(cleared automatically when current field does not match) |
| 4 | ELF | End of line flag<br>0 : No end of line flag<br>1: A line is captured by DCI |
| 3 | VSF | Vsync flag<br>0: No vsync flag<br>1: A vsync blanking detected |
| 2 | ESEF | Embedded synchronous error flag<br>0: No embedded synchronous error flag<br>1: A embedded synchronous error detected |
| 1 | OVRF | FIFO overrun flag<br>0: No FIFO overrun<br>1: A FIFO overrun occurs |
| 0 | EFF | End of frame flag<br>0: No end of frame flag<br>1: A frame is captured by DCI |

### 31.7.4. Interrupt enable register (DCI_INTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CCEIE | COFIE | F1IE | F0IE | ELIE | VSIE | ESEIE | OVRIE | EFIE |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | CCEIE | CCIR error Interrupt enable |

| | | |
|---|---|---|
| | | 0: CCIR error flag won't generate interrupt<br>1: CCIR error flag will generate interrupt |
| 7 | COFIE | CCIR change of field interrupt enable<br>0: CCIR change of field flag won't generate interrupt<br>1: CCIR change of field flag will generate interrupt |
| 6 | F1IE | CCIR field 1 interrupt enable<br>0: CCIR field 1 flag won't generate interrupt<br>1: CCIR field 1 flag will generate interrupt |
| 5 | F0IE | CCIR field 0 interrupt enable<br>0: CCIR field 0 won't generate interrupt<br>1: CCIR field 0 flag will generate interrupt |
| 4 | ELIE | End of line interrupt enable<br>0: End of line flag won't generate interrupt<br>1: End of line flag will generate interrupt |
| 3 | VSIE | Vsync interrupt enable<br>0: Vsync flag won't generate interrupt<br>1: Vsync flag will generate interrupt |
| 2 | ESEIE | Embedded synchronous error interrupt enable<br>0: Embedded synchronous error flag won't generate interrupt<br>1: Embedded synchronous error flag will generate interrupt |
| 1 | OVRIE | FIFO overrun interrupt enable<br>0: FIFO overrun won't generate interrupt<br>1: FIFO overrun will generate interrupt |
| 0 | EFIE | End of frame interrupt enable<br>0: End of frame flag won't generate interrupt<br>1: End of frame flag will generate interrupt |

### 31.7.5. Interrupt flag register (DCI_INTF)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CCEIF | COFIF | F1IF | F0IF | ELIF | VSIF | ESEIF | OVRIF | EFIF |
| | | | | | | | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | CCEIF | CCIR error interrupt flag<br>Only works in CCIR interlaced mode, error is found on the SVA or EVA codes. |
| 7 | COFIF | CCIR change of field interrupt flag<br>Only works in CCIR interlaced mode |
| 6 | F1IF | CCIR field 1 interrupt flag<br>Only works in CCIR interlaced mode(cleared automatically when current field does not match) |
| 5 | F0IF | CCIR field 0 interrupt flag<br>Only works in CCIR interlaced mode(Cleared automatically when current field does not match) |
| 4 | ELIF | End of line interrupt flag |
| 3 | VSIF | Vsync interrupt flag |
| 2 | ESEIF | Embedded synchronous error interrupt flag |
| 1 | OVRIF | FIFO overrun interrupt flag |
| 0 | EFIF | End of frame interrupt flag |

### 31.7.6. Interrupt flag clear register (DCI_INTC)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CCEFC | COFFC | F1FC | F0FC | ELFC | VSFC | ESEFC | OVRFC | EFFC |
| | | | | | | | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | CCEFC | CCIR error flag clear<br>Write 1 to clear CCIR error flag |
| 7 | COFFC | CCIR change of field flag clear<br>Write 1 to clear CCIR change of field flag |

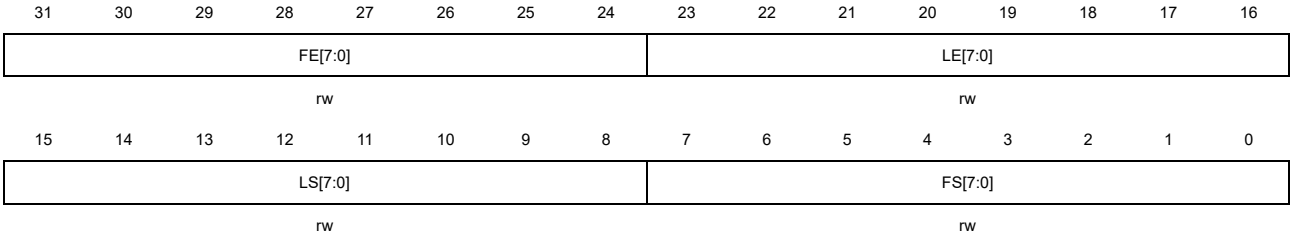
| | | |
|---|---|---|
| 6 | F1FC | CCIR field 1 interrupt flag clear<br>Write 1 to clear CCIR field 1 interrupt flag |
| 5 | F0FC | CCIR field 0 interrupt flag clear<br>Write 1 to clear CCIR field 0 interrupt flag |
| 4 | ELFC | End of line flag clear<br>Write 1 to clear end of line flag |
| 3 | VSFC | Vsync flag clear<br>Write 1 to clear vsync flag |
| 2 | ESEFC | Clear embedded synchronous error flag<br>Write 1 to clear embedded synchronous error flag |
| 1 | OVRFC | Clear FIFO overrun flag<br>Write 1 to clear FIFO overrun flag |
| 0 | EFFC | Clear end of frame flag<br>Write 1 to clear end of frame flag |

### 31.7.7. Synchronization codes register (DCI_SC)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | FE[7:0] | Frame end code in embedded synchronous mode |
| 23:16 | LE[7:0] | Line end code in embedded synchronous mode |
| 15:8 | LS[7:0] | Line start code in embedded synchronous mode |
| 7:0 | FS[7:0] | Frame start code in embedded synchronous mode |

### 31.7.8. Synchronization codes unmask register (DCI_SCUMSK)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FEM[7:0] | | | | | | | | LEM[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LSM[7:0] | | | | | | | | FSM[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | FEM[7:0] | Frame end code unmask bits in embedded synchronous mode |
| 23:16 | LEM[7:0] | Line end code unmask bits in embedded synchronous mode |
| 15:8 | LSM[7:0] | Line start code unmask bits in embedded synchronous mode |
| 7:0 | FSM[7:0] | Frame start code unmask bits in embedded synchronous mode |

### 31.7.9. Cropping window start position register (DCI_CWSPOS)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | WVSP[12:0] | | | | | | | | | | | | |
| | | | rw | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | WHSP[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:16 | WVSP[12:0] | Window vertical start position<br>Zero means the first line |
| 15:14 | Reserved | Must keep the reset value |
| 13:0 | WHSP[13:0] | Window horizontal start position<br>Zero means the first pixel clock in a line |

### 31.7.10. Cropping window size register (DCI_CWSZ)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WVSZ[13:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | WHSZ[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:16 | WVSZ[13:0] | Window vertical size<br>WVSZ=X means X+1 lines |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:0 | WHSZ[13:0] | Window horizontal size<br>WHSZ=X means X+1 pixels clock in a line |

## 31.7.11. DATA register (DCI_DATA)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DT3[[7:0] | | | | | | | | DT2[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DT1[7:0] | | | | | | | | DT0[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DT3[7:0] | Pixel data 3 |
| 23:16 | DT2[7:0] | Pixel data 2 |
| 15:8 | DT1[7:0] | Pixel data 1 |
| 7:0 | DT0[7:0] | Pixel data 0 |

# 32. TFT-LCD interface (TLI) (For GD32H737_757_759)

## 32.1. Overview

The TLI (TFT-LCD Interface) module handles the synchronous LCD interface and provides pixel data, clock and timing signals for passive LCD display. It supports a wide variety of displays with fully programmable timing parameters. A built-in DMA engine continuously move data from system memory to TLI and then, output to an external LCD display. Two separate layers are supported in TLI, as well as layer window and blending function.

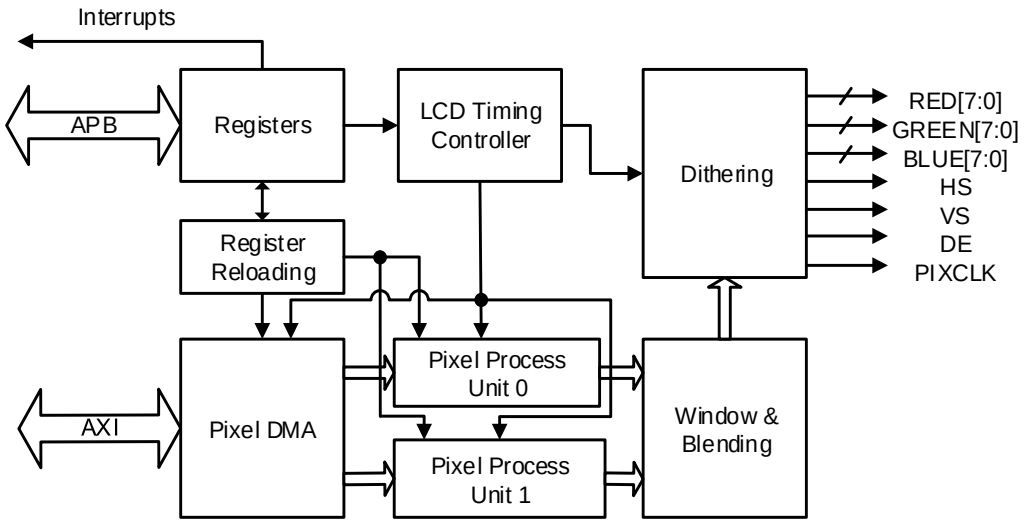
## 32.2. Characteristics

- Supports up to 24 bits data output per pixel
- Supports up to 2048 x 2048 resolution
- Timing parameters is fully programmable
- Built-in DMA engine to handle frame data copy
- 2 separate frame layers with window and blending function
- Support various pixel formats: ARGB8888, RGB888, RGB565, etc
- Support CLUT (Color Look-Up-Table) and Color-Keying format
- Dithering operation to low bits of a pixel

## 32.3. Block diagram

***Figure 32-1. TLI module block diagram*** shows the block diagram of the TLI module. There are three clock domains in TLI. The register works in APB clock and is visited by system APB bus. The Pixel DMA module works in AXI clock and fetches pixel data from system memory using AXI bus. The remaining modules work in TLI clock. The TLI clock is divided from PLL2 clock. The parameters of PLL2 and dividing factor are configured in RCU module.

**Figure 32-1. TLI module block diagram**




## 32.4. Signal description

TLI provides a 24-bit RGB Parallel display interface, which is shown in ***Table 32-1. Pins of display interface provided by TLI***.

**Table 32-1. Pins of display interface provided by TLI**

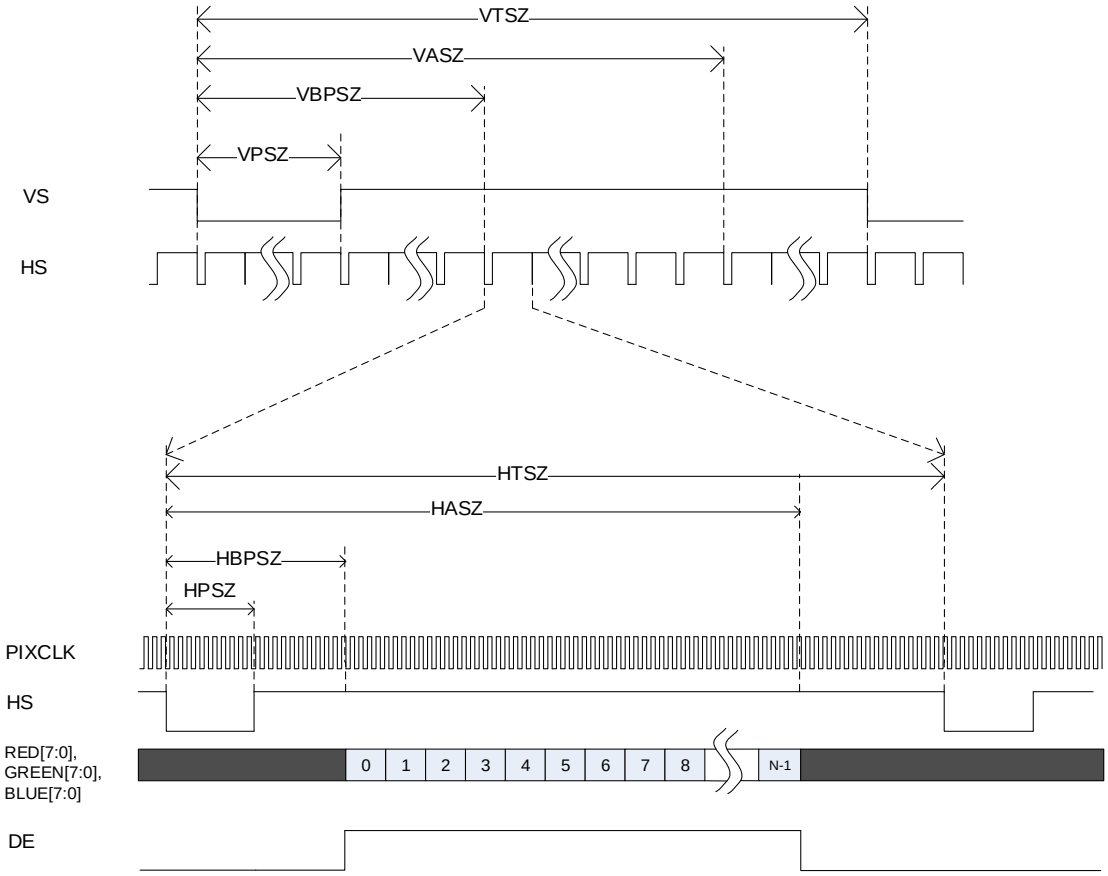
| Direction | Name | Width | Description |
|---|---|---|---|
| Output | HS | 1 | Horizontal synchronous |
| Output | VS | 1 | Vertical synchronous |
| Output | DE | 1 | Data enable |
| Output | PIXCLK | 1 | Pixel clock |
| Output | RED[7:0] | 8 | Pixel red data |
| Output | GREEN[7:0] | 8 | Pixel green data |
| Output | BLUE[7:0] | 8 | Pixel blue data |

## 32.5. Function overview

### 32.5.1. LCD display timing

LCD interface is a synchronous data interface with pixel clock, pixel data and horizontal and vertical synchronous signals. The ***Figure 32-2. Display timing diagram*** shows the signal timing of HS and VS for a whole frame. The timing parameters are configured in TLI_SPSZ, TLI_BPSZ, TLI_ASZ and TLI_TSZ registers. The timing values in these registers assume that the position of the first point is (0, 0).

**Figure 32-2. Display timing diagram**




## 32.5.2. Pixel DMA function

Following the configuration of register module, the Pixel DMA reads pixel data from memory to the pixel buffer in internal PPU (Pixel Process Unit) continuously.

After enabled, the Pixel DMA begins to fetch pixel data from system and push these data into the pixel buffer in PPU as long as the pixel buffer is not full. It always tries to use BURST16 AXI transaction to fetch pixel data.

TLI supports 2 separate frame layers and each layer has a separate frame buffer address in system. The Pixel DMA has only one AXI access interface, so it will perform round-robin arbitration between the 2 layers during pixels fetching, if both layers are enabled.

FBADD in TLI_LxFBADDR register define the frame buffer address or fetching address of each layer.

FLL in TLI_LxFLLEN defines the line length in bytes of a frame. If the length of a frame line in bytes is N, program FLL with N+7.

There may be some spacing between two frame lines in system memory and the spacing information is defined by STDOFF in TLI_LxFLLEN register. For example if the address of the first pixel in a frame line is M, and the address of the first pixel in the next frame line will be M+STDOFF. If there is no memory spacing between frame lines, just program STDOFF with FLL-3.

FTLN in TLI_LxFTLN register defines the number of lines in a frame.

### 32.5.3. Pixel formats

The Pixel DMA pushes pixel data into PPU in double word format and PPU (Pixel Process Unit) is responsible for converting various pixel formats into an internal ARGB8888 format. TLI supports up to eight pixel formats as shown in ***Table 32-2. Supported pixel formats***. The PPF[2:0] in TLI_LxPPF register defines the pixel format.

ARGB8888 format needs 8-bits data in each channel (Alpha, Red, Green and Blue), while ARGB1555 and ARGB4444 formats have fewer bits than 8 in some channels. PPU converts these formats into ARGB8888 by filling LSBs with MSBs for each channel. When processing RGB888 and RGB565 formats, PPU assumes that Alpha=255 and also fill filling LSBs with MSBs if the channel bit number less than 8.

AL88, AL44 and L8 formats are LUT (Look-Up-Table) formats. In these channels, L is the address of the look-up table. TLI has 2 internal look-up tables: one for each layer. The internal look-up table size is 256x24bits (256 entries and each entry stores a 24-bits RGB value). When processing LUT format pixel, PPU reads out an entry from the look-up table and uses this entry as the RGB value. Because the address of look-up table is 8-bit, PPU also fill LSBs with MSBs if L channel has bits less than 8. The entries in the look-up tables are uninitialized after reset, so the application should initialize the look-up table with proper value using TLI_LxLUT register before display a look-up table format layer. The TLI_LxLUT is a write-only register and a write operation to this register will write an entry to the look-up table.

Each layer is able to be configured into color keying mode. The register LxCKEY defines a RGB value. When color keying mode is enabled for a layer, PPU will compare each RGB value of each pixel in this layer with the LxCKEY and force the pixel's ARGB value to 0 if the value matches.

**Table 32-2. Supported pixel formats**

| PPF[2:0] | Pixel Format |
|---|---|
| 000 | ARGB8888 |
| 001 | RGB888 |
| 010 | RGB565 |
| 011 | ARGB1555 |
| 100 | ARGB4444 |
| 111 | AL88 |
| 101 | L8 |
| 110 | AL44 |

### 32.5.4. Layer window and blending function

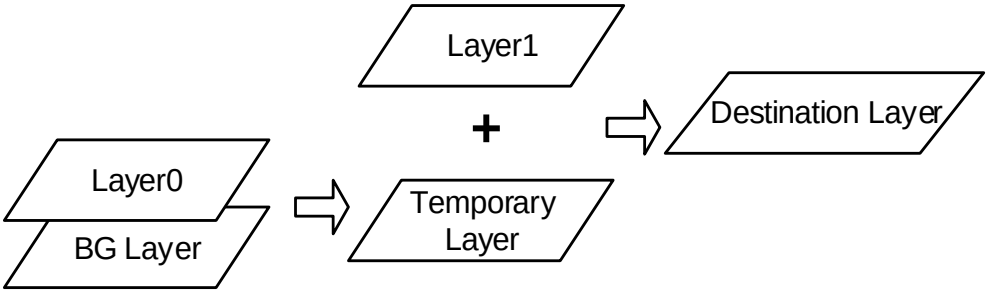
TLI supports window function for each layer and blending function between two layers. TLI first perform window operation to each layer and then blend two layers into a frame.

The window function defines a display window, and each layer has separate window

parameters defined by TLI_LxHPOS and TLI_LxVPOS registers. These window parameters define a window inside the layer. The pixel inside the window will keep its original value, while the pixel outside will be replaced with a default pixel defined in TLI_LxDC register.

The blending units first blends Layer0 and BG Layer into a temporary layer, and then blends Layer1 and the temporary layer into destination layer. BG Layer's ARGB value is defined TLI_BGC register. If a layer is disabled, blending function uses the layer's default color.

**Figure 32-3. Block diagram of Blending**




### Blending formula

The general blending formula is:

$$BC=BF_1*C+BF_2*C_S \quad (32\text{-}1)$$

- BC is blended color
- $BF_1$ is alpha calculation factor 1 of blending method
- C is current layer color
- $BF_2$ is alpha calculation factor 2 of blending method
- $C_S$ is subjacent layers blended color

The blend factor of current pixel is either normalization Pixel Alpha x normalization Specified Alpha or normalization Specified Alpha which is decided by register configuration.

## 32.5.5. Layer configuration reload

As is described above, each layer has its own frame buffer, pixel format, window, default color configuration registers and each register has a shadow register. A shadow register share the same address with the real register. Each time when the application writes to a layer-related register address, the corresponding shadow registers is updated immediately, while the real register will not change until a reload operation and only the real register has effect to the TLI function.

There are two methods for application to trigger a reload operation: request reload and frame blank reload. For request reload mode, TLI begins to load the shadow registers into real registers immediately after application set RQR bit in TLI_RL register. For frame blank reload mode, after setting FBR bit in TLI_RL register, the TLI waits for a frame vertical blanking and load the shadow registers. In both modes, hardware automatically clears the RQR or FBR bit after successfully reload.

### 32.5.6. Dithering function

The dithering module adds a 2-bit pseudo-random value to each pixel channel. This function is able to make the image smoother when 18-bits interface is used to display a 24-bit data. Application may switch on this function using DFEN bit in TLI_CTL register.

## 32.6. Interrupts

There are several status and error flags in TLI, and interrupt may be asserted from these flags. The status flags will assert global interrupt, while the error flags will assert error interrupt.

**Table 32-3. Status flags**

| Status Flag Name | Description |
|---|---|
| LMF | Line mark flag |
| LCRF | Layer configuration reloaded flag |

**Table 32-4. Error flags**

| Error Flag Name | Description |
|---|---|
| TEF | Transaction error flag |
| FEF | FIFO error flag |

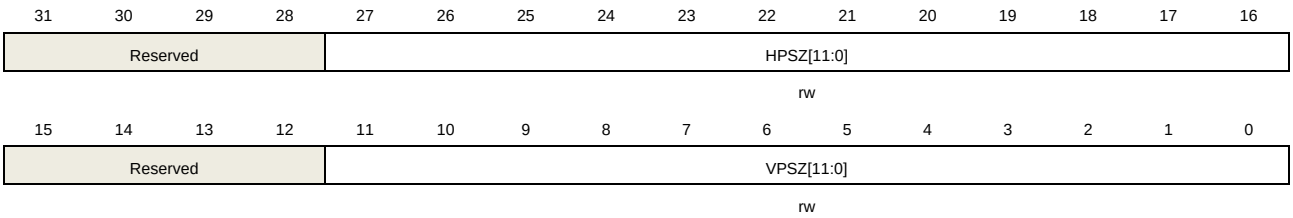
## 32.7. Register definition

TLI base address: 0x5000 1000

### 32.7.1. Synchronous pulse size register (TLI_SPSZ)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

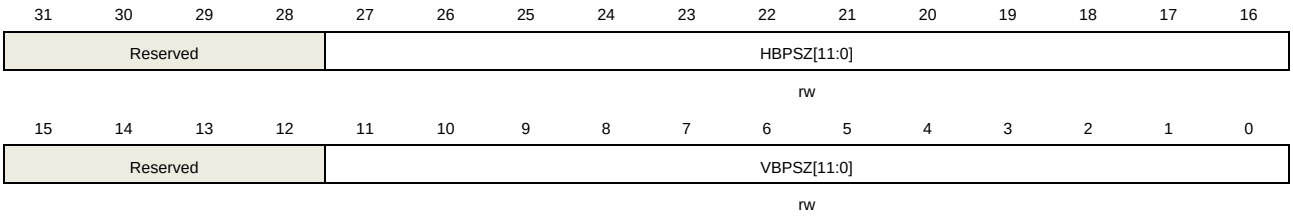



| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | HPSZ[11:0] | Size of the horizontal synchronous pulse<br>The HPSZ value should be configured to the pixels number of horizontal synchronous pulse minus 1. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | VPSZ[11:0] | Size of the vertical synchronous pulse<br>The VPSZ value should be configured to the pixels number of vertical synchronous pulse minus 1. |

### 32.7.2. Back-porch size register (TLI_BPSZ)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



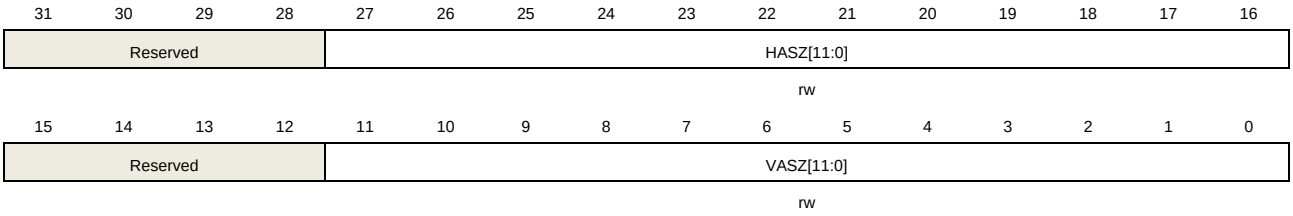

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | HBPSZ[11:0] | Size of the horizontal back porch plus synchronous pulse |

| | | |
|---|---|---|
| | | The HBPSZ value should be configured to the pixels number of horizontal back porch and synchronous pulse minus 1. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | VBPSZ[11:0] | Size of the vertical back porch plus synchronous pulse<br>The VBPSZ value should be configured to the pixels number of vertical back porch and synchronous pulse minus 1. |

### 32.7.3. Active size register (TLI_ASZ)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | HASZ[11:0] | Size of the horizontal active area width plus back porch and synchronous pulse<br>The HASZ value should be configured to the pixels number of horizontal active area width plus back porch and synchronous pulse minus 1. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | VASZ[11:0] | Size of the vertical active area width plus back porch and synchronous pulse<br>The VASZ value should be configured to the pixels number of vertical active area height plus back porch and synchronous pulse minus 1. |

### 32.7.4. Total size register (TLI_TSZ)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | HTSZ[11:0] | Horizontal total size of the display, including active area, back porch, synchronous pulse and front porch<br>The HTSZ value should be configured to the pixels number of horizontal active area width plus back porch, front porch and synchronous pulse minus 1. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | VTSZ[11:0] | Vertical total size of the display, including active area, back porch, synchronous pulse and front porch<br>The VTSZ value should be configured to the pixels number of vertical active area height plus back porch, front porch and synchronous pulse minus 1. |

### 32.7.5. Control register (TLI_CTL)

Address offset: 0x18

Reset value: 0x0000 2220

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HPPS | VPPS | DEPS | CLKPS | Reserved | | | | | | | | | | | DFEN |
| rw | rw | rw | rw | | | | | | | | | | | | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RDB[2:0] | | | Reserved | GDB[2:0] | | | Reserved | BDB[2:0] | | | Reserved | | | TLIEN |
| | r | | | | r | | | | r | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | HPPS | Horizontal pulse polarity selection<br>0: Horizontal synchronous pulse active low<br>1: Horizontal synchronous pulse active high |
| 30 | VPPS | Vertical pulse polarity selection<br>0: Vertical synchronous pulse active low<br>1: Vertical synchronous pulse active high |
| 29 | DEPS | Not data enable polarity selection<br>0: Not data enable active low<br>1: Not data enable active high |
| 28 | CLKPS | Pixel clock polarity selection<br>0: Pixel clock is TLI clock<br>1: Pixel clock is inverted TLI clock |
| 27:17 | Reserved | Must be kept at reset value. |
| 16 | DFEN | Dither function enable<br>0: Dither function disable |

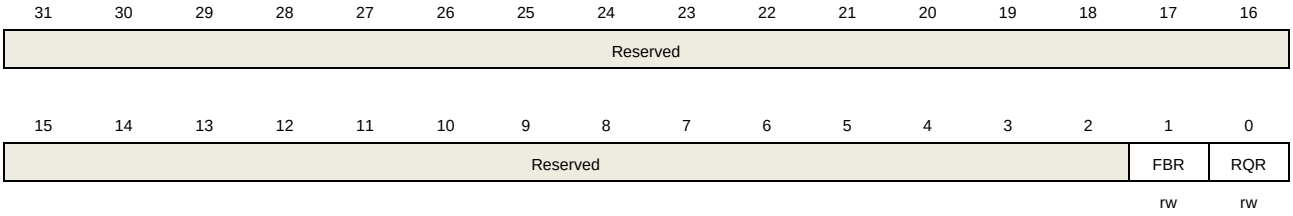
| | | |
|---|---|---|
| | | 1: Dither function enable |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | RDB[2:0] | Red channel dither bits number<br>Fixed to 2, read only |
| 11 | Reserved | Must be kept at reset value. |
| 10:8 | GDB[2:0] | Green channel dither bits number<br>Fixed to 2, read only |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | BDB[2:0] | Blue channel dither bits number<br>Fixed to 2, read only |
| 3:1 | Reserved | Must be kept at reset value. |
| 0 | TLIEN | TLI enable bit<br>0: TLI disable<br>1: TLI enable |

### 32.7.6. Reload layer register (TLI_RL)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



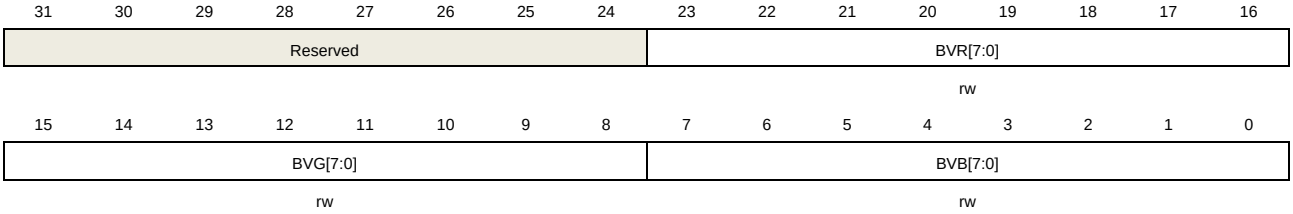

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | FBR | Frame blank reload<br>This bit is set by software and cleared by hardware after reloading<br>0: Reload disable<br>1: The layer configuration will be reloaded into core at frame blank |
| 0 | RQR | Request reload<br>This bit is set by software and cleared by hardware after reloading<br>0: Reload disable<br>1: The layer configuration will be reloaded into core after this bit sets |

### 32.7.7. Background color register (TLI_BGC)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



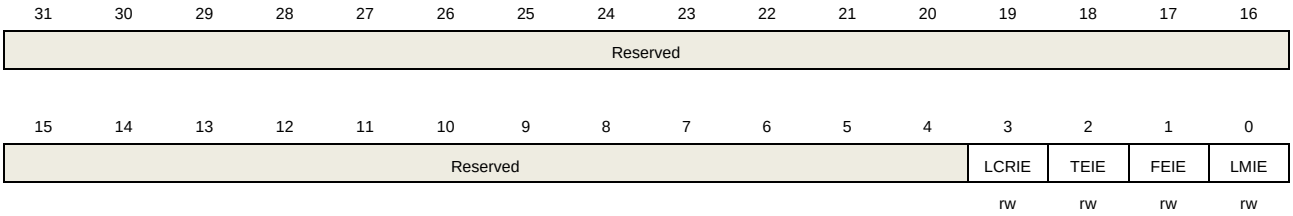

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | BVR[7:0] | Background value red |
| 15:8 | BVG[7:0] | Background value green |
| 7:0 | BVB[7:0] | Background value blue |

### 32.7.8. Interrupt enable register (TLI_INTEN)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | LCRIE | Layer configuration reloaded interrupt enable<br>0: Layer configuration reloaded flag won't generate an interrupt<br>1: Layer configuration reloaded flag will generate an interrupt |
| 2 | TEIE | Transaction error interrupt enable<br>0: Transaction error flag won't generate an interrupt<br>1: Transaction error flag will generate an interrupt |
| 1 | FEIE | FIFO error interrupt enable<br>0: FIFO error flag won't generate an interrupt<br>1: FIFO error flag will generate an interrupt |

| | | |
|---|---|---|
| 0 | LMIE | Line mark interrupt enable<br>0: Line mark flag won't generate an interrupt<br>1: Line mark flag will generate an interrupt |

## 32.7.9. Interrupt flag register (TLI_INTF)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | LCRF | TEF | FEF | LMF |
| | | | | | | | | | | | | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | LCRF | Layer configuration reloaded flag<br>0: No layer configuration reloaded flag<br>1: Layer configuration is reloaded triggered by FBR bit in TLI_RL |
| 2 | TEF | Transaction error flag<br>0: No transaction error flag<br>1: A transaction error on AHB bus occurs |
| 1 | FEF | FIFO error flag<br>0: No FIFO error flag<br>1: A FIFO under-run error occurs<br>The under-run error occurs when the value written in TLI_LxFLLEN and TLI_LxFTLN is less than required. |
| 0 | LMF | Line mark flag<br>0: No line mark flag<br>1: Line number reaches the specified value in TLI_LM |

## 32.7.10. Interrupt flag clear register (TLI_INTC)

Address offset: 0x3C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | LCRC | TEC | FEC | LMC |
|---|---|---|---|---|
| | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | LCRC | Layer configuration reloaded flag clear<br>Write 1 to clear layer configuration reloaded flag |
| 2 | TEC | Transaction error flag clear<br>Write 1 to clear transaction error flag |
| 1 | FEC | FIFO error flag clear<br>Write 1 to clear FIFO error flag |
| 0 | LMC | Line mark flag clear<br>Write 1 to clear line mark flag |

## 32.7.11. Line mark register (TLI_LM)

Address offset: 0x40

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | LM[10:0] | | | | | | | | | | |
| | | | | | rw | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10:0 | LM[10:0] | Line mark value<br>The LMF bit in TLI_INTF will be set after the line number reaches this value |

## 32.7.12. Current pixel position register (TLI_CPPOS)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HPOS[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VPOS[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

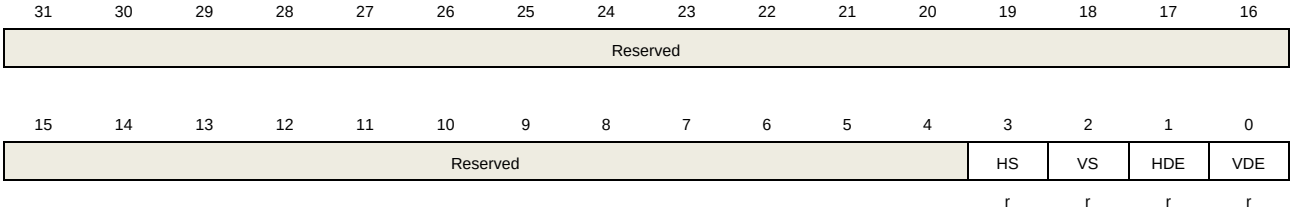
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | HPOS[15:0] | Horizontal position<br>Horizontal position of the current displayed pixel |
| 15:0 | VPOS[15:0] | Vertical position<br>Vertical position of the current displayed pixel |

### 32.7.13. Status register (TLI_STAT)

Address offset: 0x48

Reset value: 0x0000 000F

This register has to be accessed by word (32-bit).



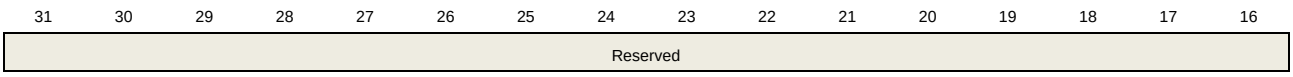

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3 | HS | Current HS status of the TLI |
| 2 | VS | Current VS status of the TLI |
| 1 | HDE | Current HDE status<br>0: HPOS in TLI_CPPOS register is not between the HBPSZ in TLI_BPSZ register and HASZ in TLI_ASZ register.<br>1: HPOS in TLI_CPPOS register is between the HBPSZ in TLI_BPSZ register and HASZ in TLI_ASZ register. |
| 0 | VDE | Current VDE status<br>0: VPOS in TLI_CPPOS register is not between the VBPSZ in TLI_BPSZ register and VASZ in TLI_ASZ register.<br>1: VPOS in TLI_CPPOS register is between the VBPSZ in TLI_BPSZ register and VASZ in TLI_ASZ register. |

### 32.7.14. Layer x control register (TLI_LxCTL) (x = 0,1)

Address offset: 0x84+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | LUTEN | Reserved | | CKEYEN | LEN |
| | | | | | | | | | | | rw | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | LUTEN | LUT enable<br>0: LUT is disabled<br>1: LUT is enabled |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | CKEYEN | Color keying enable<br>0: Color keying is disabled<br>1: Color keying is enabled |
| 0 | LEN | Layer enable<br>0: This layer is disabled<br>1: This layer is enabled |

## 32.7.15. Layer x horizontal position parameters register (TLI_LxHPOS) (x = 0,1)

Address offset: 0x88+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | WRP[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | WLP[11:0] | | | | | | | | | | | |
| | | | | rw | | | | | | | | | | | |

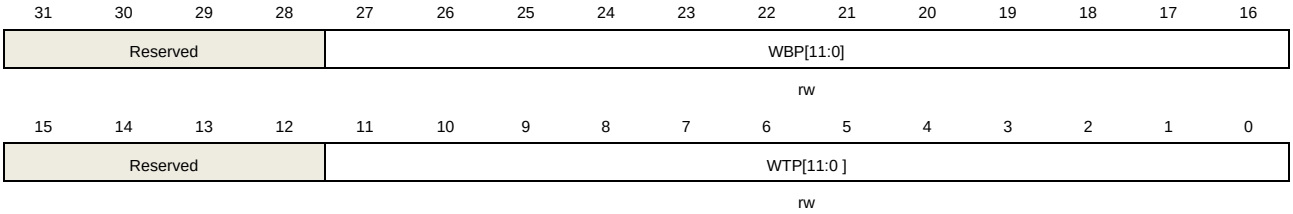
| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | WRP[11:0] | Window right position |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | WLP[11:0] | Window left position |

## 32.7.16. Layer x vertical position parameters register (TLI_LxVPOS) (x = 0,1)

Address offset: 0x8C+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

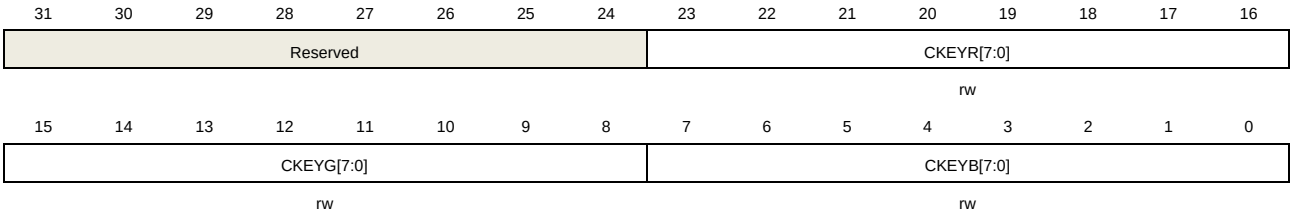



| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:16 | WBP[11:0] | Window bottom position |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:0 | WTP[11:0] | Window top position |

## 32.7.17. Layer x color key register (TLI_LxCKEY) (x = 0,1)

Address offset: 0x90+0x80 * x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



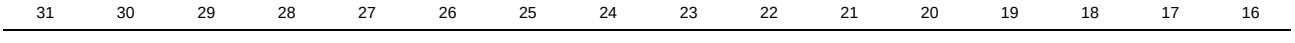

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | CKEYR[7:0] | Color key red |
| 15:8 | CKEYG[7:0] | Color key green |
| 7:0 | CKEYB[7:0] | Color key blue |

If the pixel RGB value in a layer equals the value in TLI_LxCKEY, the pixel RGB value is reset to 0. That means these pixels is transparent to other layers.

## 32.7.18. Layer x packeted pixel format register (TLI_LxPPF) (x = 0,1)

Address offset: 0x94+0x80 * x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | PPF[2:0] | |
| | | | | | | | | | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2:0 | PPF[2:0] | Packeted pixel format<br>These bits configures the packeted pixel format<br>000: ARGB8888<br>001: RGB888<br>010: RGB565<br>011: ARGB1555<br>100: ARGB4444<br>101: L8<br>110: AL44<br>111: AL88 |

## 32.7.19. Layer x specified alpha register (TLI_LxSA) (x = 0,1)

Address offset: 0x98+0x80 * x

Reset value: 0x0000 00FF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | SA[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | SA [7:0] | Specified alpha<br>The alpha value used for blending |

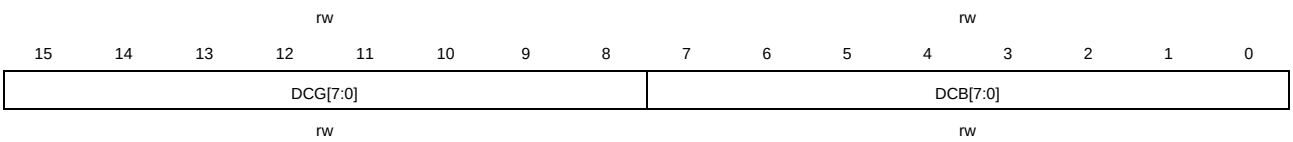
## 32.7.20. Layer x default color register (TLI_LxDC) (x = 0,1)

Address offset: 0x9C+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DCA[7:0] | | | | | | | | DCR[7:0] | | | | | | | |



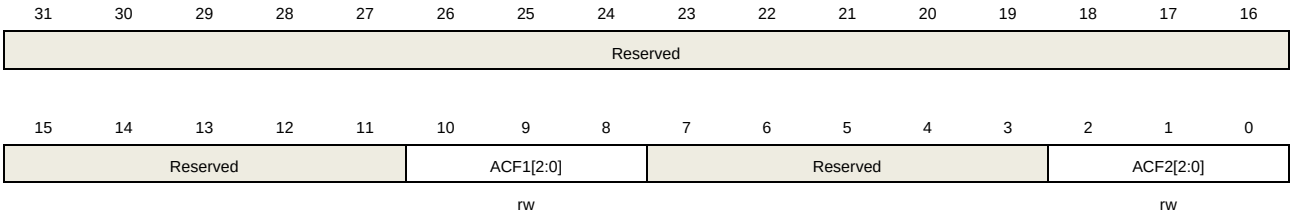

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DCA[7:0] | The default color alpha |
| 23:16 | DCR[7:0] | The default color red |
| 15:8 | DCG[7:0] | The default color green |
| 7:0 | DCB[7:0] | The default color blue |

The default color of a layer takes effect when the layer is disabled or outside the window defined in TLI_LxHPOS and TLI_LxVPOS.

## 32.7.21. Layer x blending register (TLI_LxBLEND) (x = 0,1)

Address offset: 0xA0+0x80 * x
Reset value: 0x0000 0607

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10:8 | ACF1[2:0] | Alpha calculation factor 1 of blending method<br>000: Reserved<br>001: Reserved<br>010: Reserved<br>011: Reserved<br>100: normalization Specified Alpha<br>101: Reserved<br>110: normalization Pixel Alpha x normalization Specified Alpha<br>111:Reserved |
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | ACF2[2:0] | Alpha calculation factor 2 of blending method<br>000: Reserved<br>001: Reserved<br>010: Reserved |

011: Reserved

100: Reserved

101:1- normalization specified alpha

110: Reserved

111: 1-normalization pixel alpha x normalization specified alpha

## 32.7.22. Layer x frame base address register (TLI_LxFBADDR) (x = 0,1)

Address offset: 0xAC+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FBADD[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FBADD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FBADD[[31:0] | Frame buffer base address<br>The base address of frame buffer |

## 32.7.23. Layer x frame line length register (TLI_LxFLLEN) (x = 0,1)

Address offset: 0xB0+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | STDOFF[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FLL[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

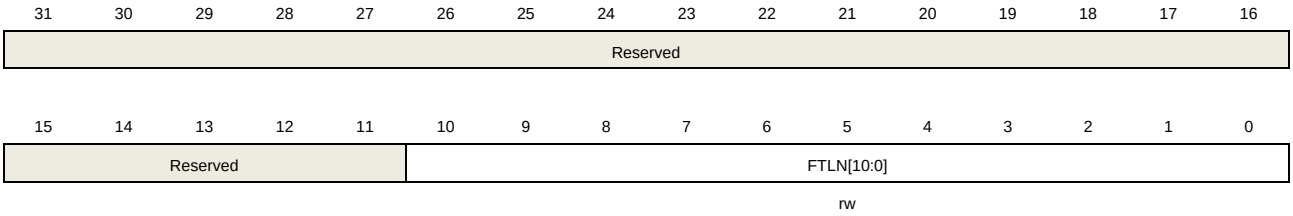
| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:16 | STDOFF[13:0] | Frame buffer stride offset<br>This value defines the bytes number from start of a line to the start of next line |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:0 | FLL[13:0] | Frame line length<br>This value defines the bytes number of a line plus 7 |

### 32.7.24. Layer x frame total line number register (TLI_LxFTLN) (x = 0,1)

Address offset: 0xB4+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



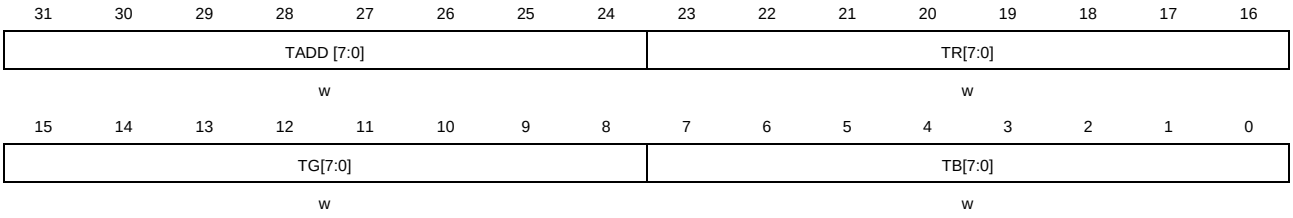

| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10:0 | FTLN[10:0] | Frame total line number<br>This value defines the line number in a frame |

### 32.7.25. Layer x look up table register (TLI_LxLUT) (x = 0,1)

Address offset: 0xC4+0x80 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | TADD[7:0] | Look up table write address<br>The entry at this address in LUT will be updated with the value of red, green and bule written |
| 23:16 | TR[7:0] | Red channel of a LUT entry |
| 15:8 | TG[7:0] | Green channel of a LUT entry |
| 7:0 | TB[7:0] | Blue channel of a LUT entry |

# 33. Receiver of Sony/Philips Digital Interface (RSPDIF) (For GD32H737_757_759)

## 33.1. Overview

The receiver of Sony/Philips Digital Interface (RSPDIF) module provides the function of receiving and decoding RSPDIF audio data streams.

## 33.2. Characteristics

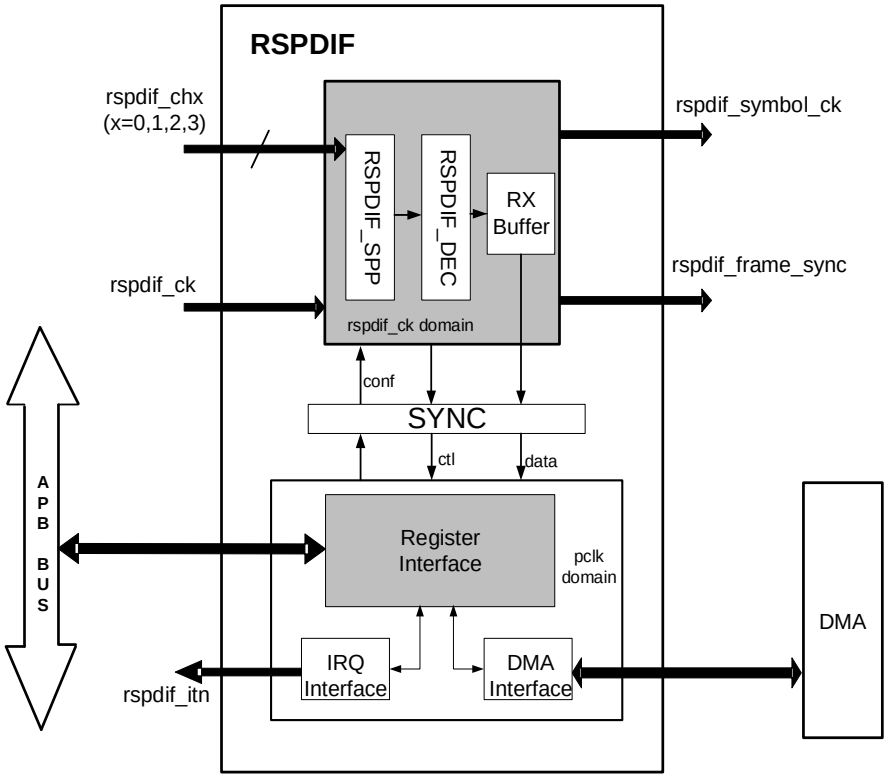
- Supports audio IEC-60958 and IEC-61937.
- Up to 4 inputs available.
- Supports receiving double buffer.
- Dual clock domain：PCLK for register interface and RSPDIF_CK for other parts.
- Supports maximum symbol rate: 12.288 MHz.
- Supports stereo stream from 8 to 192 kHz.
- Supports automatic symbol rate detection.
- Generate symbol clock.
- Check the parity bit of the received data.
- Support multiple data processing methods, which can process audio data and user channel information separately or together.
- Supports using DMA communication to receive audio data and user channel information respectively.
- Supports interrupt function.

## 33.3. Function overview

### 33.3.1. RSPDIF block diagram

The receiver of Sony/Philips Digital Interface (RSPDIF) module provides the function of receiving and decoding SPDIF audio data streams.The audio data, channel status (CS) and User data (U) received can be accessed by DMA interface. The RSPDIF transceiver allows the handling of both RSPDIF channel status (CS) and User data (U) and supports the precise measurement of an incoming sampling frequency.

**Figure 33-1. RSPDIF block diagram**




**Legend:**

rspdif_frame_sync: RSPDIF frame rate synchronization signal

rspdif_symbol_ck: RSPDIF channel symbol clock

The RSPDIF module is responsible for decoding the S/PDIF stream received from RSPDIF_CH[3:0], and is used to receive S/PD audio data in accordance with the IEC-60958 and IEC-61937 standards. These standards support simple stereo streams at high sampling rates, as well as compressed multi-channel surround sound.

RSPDIF resamples the incoming signal, decodes the Manchester stream, recognizes subframes, frames, and block elements, and passes the decoded data and associated status flags to the RSPDIF register. The RSPDIF section can be fully controlled through the APB bus, and provides a dedicated path for user information and channel information. It can handle two DMA channels, audio sample and channel/user information. Interrupt service can also be used as an alternative to DMA for sending error signals or critical states.

When RSPDIF decodes the incoming audio data stream, it also provides two signals:

- rspdif_frame_sync: The frequency of this signal is equal to the frame rate, and it is inverted every time the subframe preamble is detected by RSPDIF, and the duty cycle is 50%.
- rspdif_symbol_ck: The signal frequency is equal to the symbol rate. The signal can drive internal components in the system, such as SAI ports, and external components, such as A/Ds or D/As, with clocking control provided via related registers. Please refer to

*RSPDIF clock management* for specific generation conditions.

### 33.3.2. S/PDIF protocol

S/PDIF (Sony/Philips Digital Interface) is a type of digital audio interconnect used in consumer audio equipment to output audio over reasonably short distances. The signal is transmitted over either a coaxial cable with RCA connectors or a fiber optic cable with TOSLINK connectors. S/PDIF interconnects components in home theaters and other digital high-fidelity systems.

S/PDIF is a data link layer protocol as well as a set of physical layer specifications for carrying digital audio signals between devices and components over either optical or electrical cable. The name stands for Sony/Philips Digital Interconnect Format but is also known as Sony/Philips Digital Interface. Sony and Philips were the primary designers of S/PDIF. S/PDIF is standardized in IEC 60958.

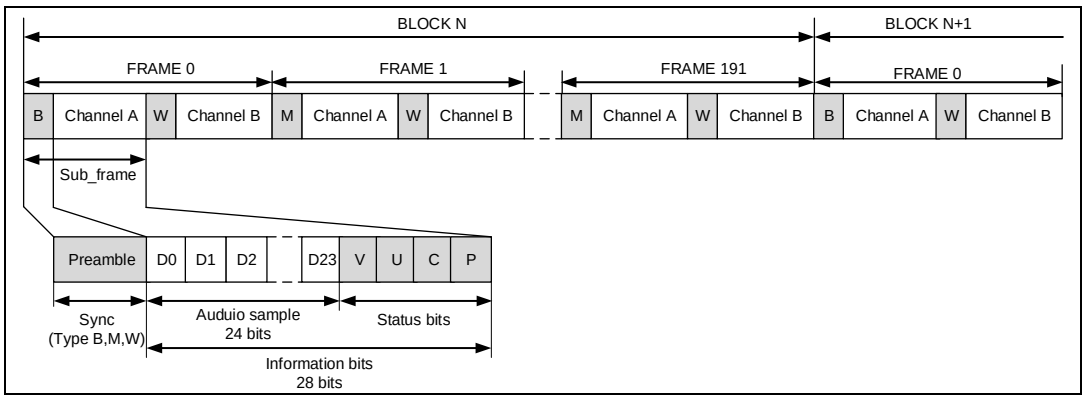
#### S/PDIF block

A S/PDIF block consists of 192 frames and each frame contains two sub-frames(one for left channel and the other one for right channel), that is, a RSPDIF block contains 384 sub-frames. Each sub-frame contains 32 bits. The data format of the block and the sub-frame is shown in *Table 33-1. Sub-frame pattern* and *Figure 33-2. S/PDIF block and S/PDIF sub-frame format.*

**Table 33-1. Sub-frame pattern**

| BITS | Description |
|---|---|
| 0~3 | synchronization preambles, three types: B, M, W.<br>Preambe M represents channel A (left channel) or main is transmitted at this time, preamble W represents channel B (right channel) is transmitted at this time, and preamble B is more special, which represents channel A (left channel) is transmitted at this time, and it is the starting subframe of A block. |
| 4~27 | the audio sample word in linear 2's complement representation. The most significant bit (MSB) is always carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8. |
| 28 | validity bit “V”, indicates if the data is valid |
| 29 | user data bit “U”, defined by user |
| 30 | channel status bit “C” |
| 31 | parity bit “P”, carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity). |

**Figure 33-2. S/PDIF block and S/PDIF sub-frame format**

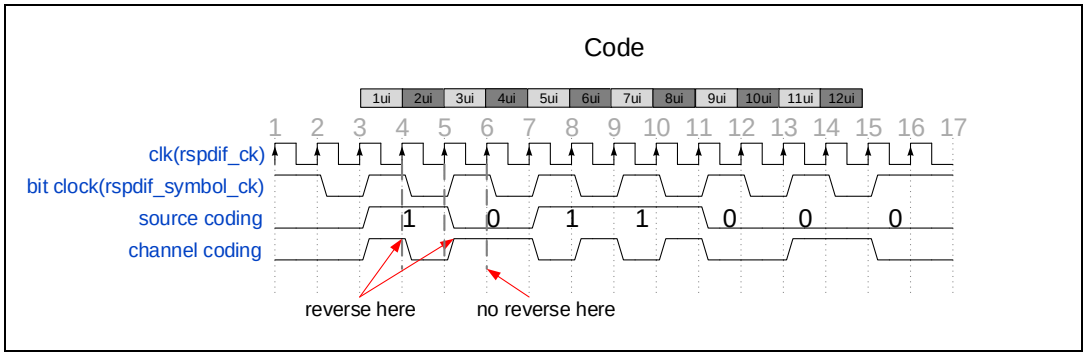



## Coding of information bit

When IEC60958 transmits data, bits 4 to 31 are encoded in biphase-mark(Manchester protocol). The principle is to use a clock frequency that is twice the transmission bit rate as a reference, split the original one bit of data into two, and each bit to be transmitted is represented by a symbol containing two consecutive binary states. When the data is 1, in the clock cycle shift a potential (0->1 or 1->0) data into two different potentials, into 10 or 01, When the data is 0, it doesn't switch, it goes to 11 or 00. These binary states are named “UI” (Unit Interval) in the IEC-60958 specification.

**Note:** A symbol represents a data bit which comprises two unit intervals. The symbols rate means the frequency of bit clock shown in ***Figure 33-3. Information bits coding example.*** The clock frequecny used by RSPDIF module is twice of that of bit clock.

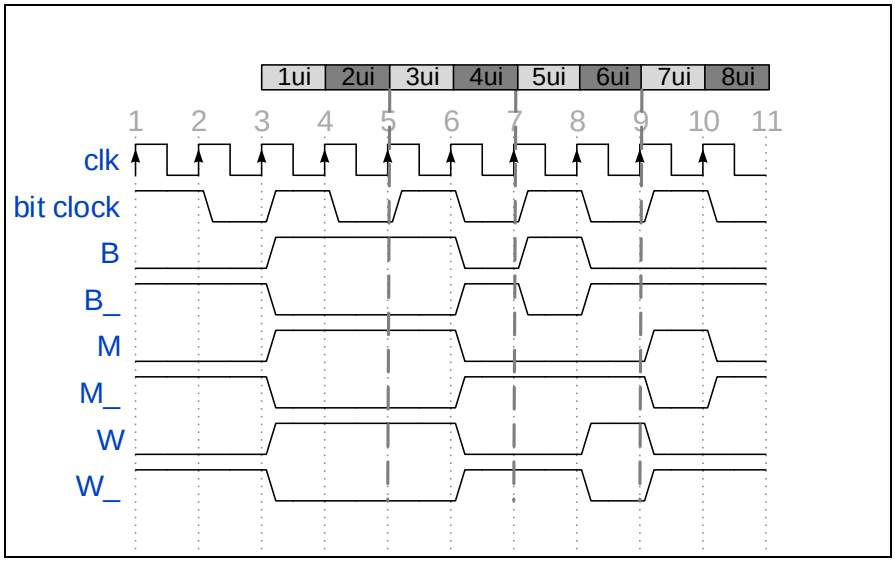
**Figure 33-3. Information bits coding example**




## Synchronization preambles

Whether or not to invert the preamble pattern is determined by the value of the previous half-bit value. For the first "B" preamble of the first frame, this first half-bit value is the line level before transmission is enabled. For other preambles, this first half-bit value is the second half of the parity bit of the previous sub-frame. The preambles patterns B, M and W are shown in the ***Figure 33-4. S/PDIF Preambles.*** The preamble pattern can be concluded in ***Table 33-2. Preamble pattern***.

**Figure 33-4. S/PDIF Preambles**




**Note:** When coding data, transition should occur at the end of second UI. However, transition may lack in preamble at the end of second UI.

**Table 33-2. Preamble pattern**

| Preceding state (the previous half-bit value) | 0 | 1 | Description |
|---|---|---|---|
| Preamble | Channel coding | | |
| B | 11101000 | 00010111 | Channel A data at the start of block |
| W | 11100100 | 00011011 | Channel B data at somewhere in the block |
| M | 11100010 | 00011101 | Channel A data |

### 33.3.3. RSPDIF

RSPDIF mainly includes two parts: RSPDIF_SPP and RSPDIF_DEC. The S/PDIF stream is decoded based on measuring the time interval between two consecutive edges. There are three kinds of time intervals in the S/PDIF stream, shown as ***Table 33-3. RSPDIF time interval***.

**Table 33-3. RSPDIF time interval**

| Time interval | Description |
|---|---|
| TL | long time interval, having a duration of 3 x UI. It appears only during preambles. |
| TM | medium time interval, having a duration of 2 x UI. It appears both in some preambles or the information field. |
| TS | short time interval, having a duration of 1 x UI. It appears both in some preambles or the information field. |

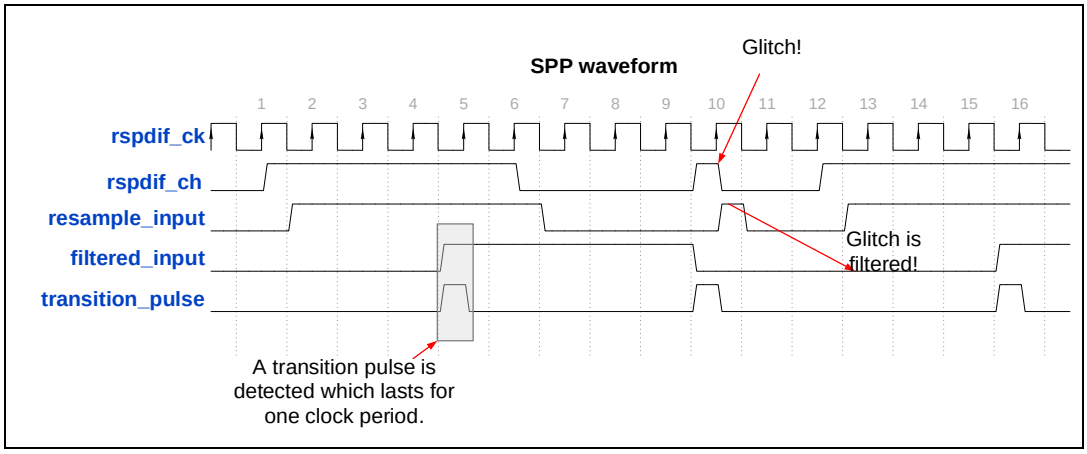
### RSPDIF_SPP

RSPDIF_SPP signal preprocessing module is designed to perform noise filtering and rising/falling edge detection.

RSPDIF supports a total of four input signals, select the required input by configuring the RXCHSEL[2:0] bits in RSPDIF_CTL register. The S/PDIF signal received on the selected RSPDIF_CH is re-sampled using the rspdif_ck clock (acquisition clock). A simple filtering is applied in order to eliminate glitches.

This is performed by the stage detecting the edge transitions. During edge conversion detection, when the sequence 0 is followed by two 1s, the rising edge is detected. When the sequence 1 is followed by two 0s, the falling edge is detected. After the rising edge, a sequence of falling edges is expected. After the falling edge, a sequence of rising edges is expected.

**Figure 33-5. Noise filtering and rising/falling edge detection**




**Note:** The transition_pulse is an important indictor for the RSPDIF_DEC module to judge the transition types and decode the input bit stream proprly.

### RSPDIF_DEC

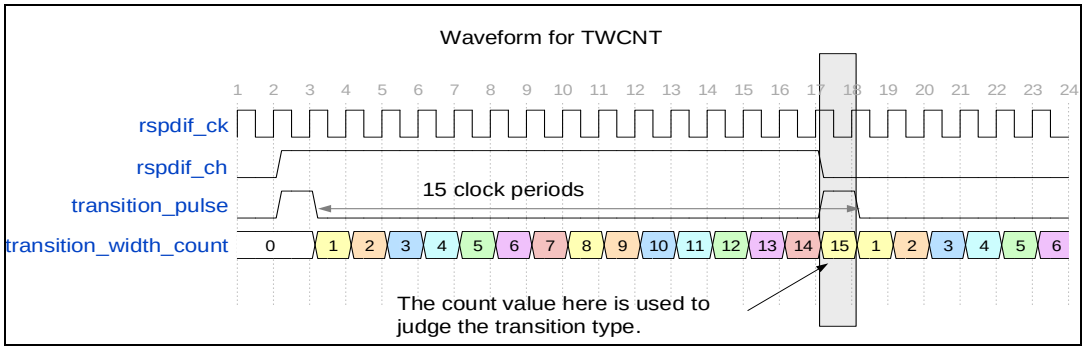
RSPDIF_DEC detects preamble type and decodes information bits. RSPDIF_DEC will pack these data and write into the RX buffer or into RSPDIFCHSTAT register. The R SPDIF_DEC module can perform functions such as time interval estimation, symbol ra te and synchronization estimation, block and subframe header detection, data decodin g and continuous symbol rate tracking.

#### TWCNT

The TWCNT counter is used to measure the time interval duration. It is clocked by the rspdif_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again. If the time interval between the two transitions is too long and the TWCNT overflows, RSPDIF will stop working and the TMOUTERR flag of the RSPDIF_STAT register is set at this time.

The transition timer is like a watchdog timer, which generates a trigger after 70 conversions of the incoming signal, and stops searching for the longest and shortest conversions. Note that calculating 70 transitions ensures a delay that is a little longer than a subframe.

**Figure 33-6. Waveforms of TWCNT**




***Note:*** The counter value stored is equal to the actual clock periods between the two transition pulses. For example, as shown in ***Figure 33-6. Waveforms of TWCNT***, the time interval duaration between the two transition pulses is 15 RSPDIF_CK clock periods, so the number stored is 15(not 14!). In this way, it is convient to calculate CKCNT5, WIDTH24 and WIDTH40. When the counter value is reset, the reset value is (NOT 0!).

**SYNC**

RSPDIF_DEC provides a SYNC module, please refer to ***RSPDIF synchronization process***. SYNC module calculates WIDTH24 and WIDTH40 which are references for calculating the thresholds for judging the transition type. WIDTH24 and WIDTH40 represent the time interval duaration of consective 24 symbols and 40 symbols respectively. As introduced above, it is clear that:

$$WIDTH24 = 48\ UI \rightarrow TL_{LO} = 1.5\ UI = \frac{WIDTH24}{32} \quad (33\text{-}1)$$

$$WIDTH40 = 80\ UI \rightarrow TH_{HI} = 2.5\ UI = \frac{WIDTH40}{32} \quad (33\text{-}2)$$

$TH_{HO}$ and $TL_{LO}$ is threshold values which are used for judging the transition types and this part shall be introduced in the description of DECODER module and RSPDIF synchronization process in detail. WIDTH24 and WIDTH40 can also be used for generating the symbol clock and this will be introduced ***RSPDIF clock management***.

Additionally, CKCNT5 in RSPDIF_STAT register which is used to estimate the S/PDIF symbol rate is also calculated in SYNC.

**DECODER**

RSPDIF_DEC provides a DECODER module which has the function of transition coder and preamble detector. It outputs the preamble type (PREF), 28 bit information bits and status bits (C,U,V,P). A recovered symbol clock is also generated during the decoder process of the RSPDIF_CH bit stream. Decoder block receives the current transition width from the TWCNT counter. This block encodes the current transition width by comparing the current transition

width with two different thresholds, names $TH_{HI}$ and $TH_{LO}$, as shown in ***Table 33-4. Transition encoder encoding rules***.

**Table 33-4. Transition encoder encoding rules**

| transition width (TH) | Code | Description |
|---|---|---|
| TH < ($TH_{LO}$ – 1) | TS | the data received is half part of data bit '1' |
| ($TH_{LO}$ – 1)< TH <$TH_{HI}$ | TM | the data received is data bit '0' |
| TH > $TH_{HI}$ | TL | the data received is the long pulse of preambles |
| else | - | FRERR flag is set |

The calculation methods of THHI and THLO, refer to ***Table 33-5. Thresholds calculation***.

**Table 33-5. Thresholds calculation**

| Thresholds | Synchronization |
|---|---|
| $TH_{LO}$ | WIDTH24 / 32 |
| $TH_{HI}$ | WIDTH40 / 32 |

RSPDIF synchronization refer to ***Figure 33-7. Synchronization flowchart flow*** for additional information.

### Preamble detector

The decoder provides the function of preamble detection, checking four consecutive conversions of a specific sequence to ensure whether they constitute the preamble part. Assume that TRANS0, TRANS1, TRANS2, and TRANS3 represent the four consecutive transformations of the above encoding. The four converted values are shown in ***Table 33-6. Preamble transition sequence***. The absence of this mode indicates that these conversions do not constitute a header, but rather constitute part of the subframe data and can be decoded by a bi-phase decoder.

**Table 33-6. Preamble transition sequence**

| Preamble type | Biphase data pattern | TRANS3 | TRANS2 | TRANS1 | TRANS0 |
|---|---|---|---|---|---|
| B | 11101000 | TL | TS | TS | TL |
| M | 11100010 | TL | TL | TS | TS |
| W | 11100100 | TL | TM | TS | TM |

### Bi-phase decoder

The Bi-phase decoder uses the conversion information provided by the conversion encoder and the preamble detector to decode the input bi-phase symbol data stream. When the header has been correctly detected, the Bi-phase decoder will decode the following data information, as shown in ***Table 33-7. Bi-phase decoding rules***.

**Table 33-7. Bi-phase decoding rules**

| Input | Decode value |
|---|---|
| TM | 0 |
| Two consecutive TS | 1 |
| else | FRERR flag is set |

### 33.3.4. RSPDIF synchronization process

When the RXCFG[1:0] in RSPDIF_CTL register is set to 2'b01 or 2'b11, the synchronization process starts. If the WFRXA bit in RSPDIF_CTL register is set to 1, RSPDIF will first detect the activity of the selected RSPDIF_CH line before the synchronization. Only when four transitions are detected on the selected RSPDIF_CH line, RSPDIF will switch to the synchronization phase. This function can avoid synchronization errors effectively. The synchronization process is shown in ***Figure 33-7. Synchronization flowchart flow***. This function is implemented in the RSPDIF_DEC module.

After the synchronization is correctly decoded by RSPDIF, the accurate synchronization threshold is calculated. When RSPDIF can correctly measure the duration of 24 and 40 consecutive symbols, the synchronization is completed, the threshold will be updated, and the SYNDO flag is set to 1.

There may be interference on the RSPDIF_CH line, and it may happen that the synchronization process is not performed correctly. RSPDIF provides a function to set the number of synchronization retries (MAXRT) before synchronization. Until the times set by MAXRT bit are reached, the synchronization has not been performed correctly, and the SYNERR error flag is set. If there is no valid RSPDIF data stream on the RSPDIF_CH line, TWCNT overflows and the TMOUTERR error flag is set.

After synchronization is completed, when the next header "B" is detected, RSPDIF starts to receive channel status (C) and user data (U) (see ***Figure 33-8. Synchronization process scheduling***). The user reads the C and U bits through the RSPDIF_CHSTAT register, and configures the RXDF[1:0] and RXSTEOMEN bits according to the C and U bits.

**Note:** When RXCFG[1:0] = 2'b11, the modification of RXDF[1:0] and RXSTEOMEN bits is invalid. Refer to ***RSPDIF_CTL register access permissions in different states.***

**Figure 33-7. Synchronization flowchart flow**

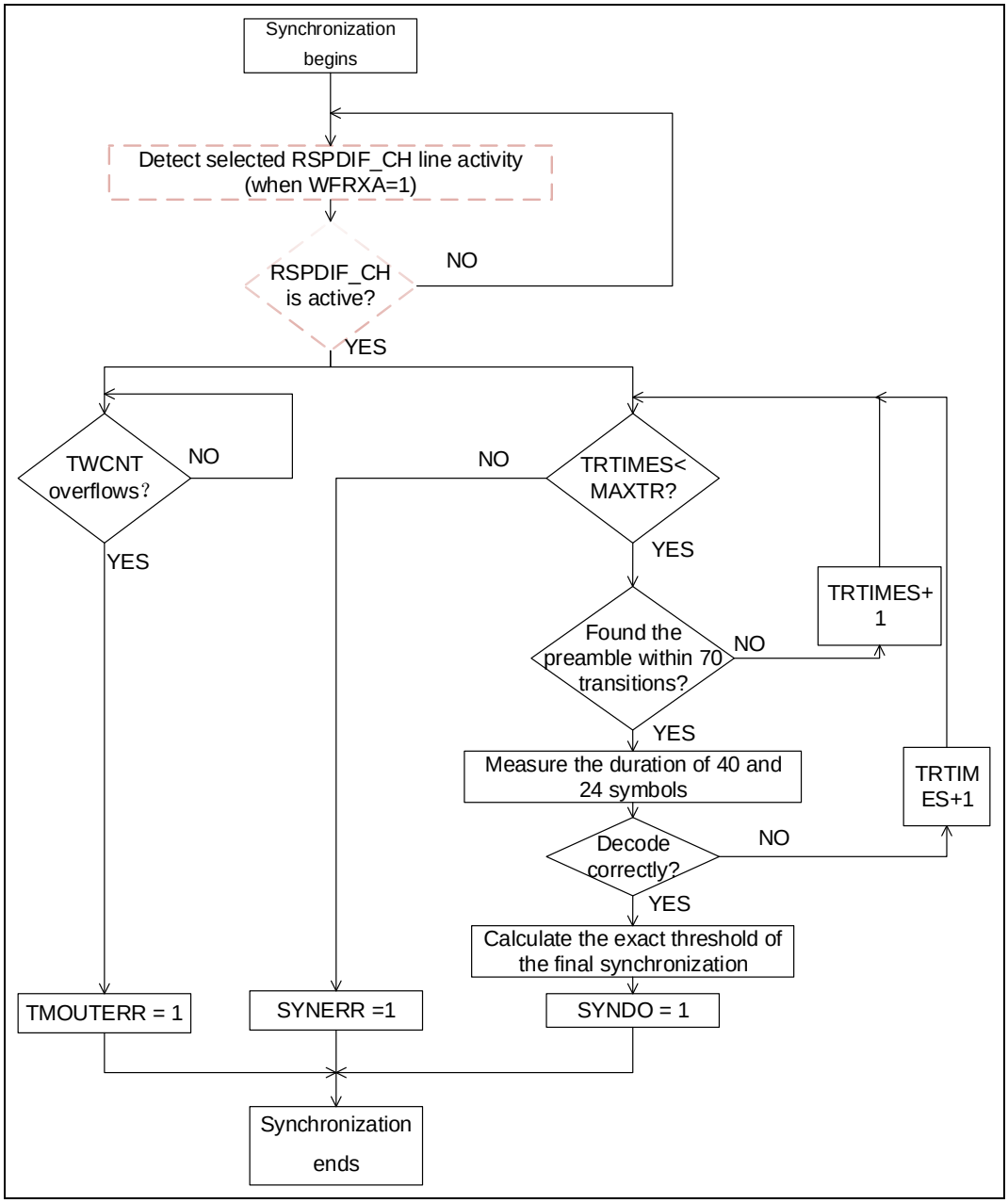
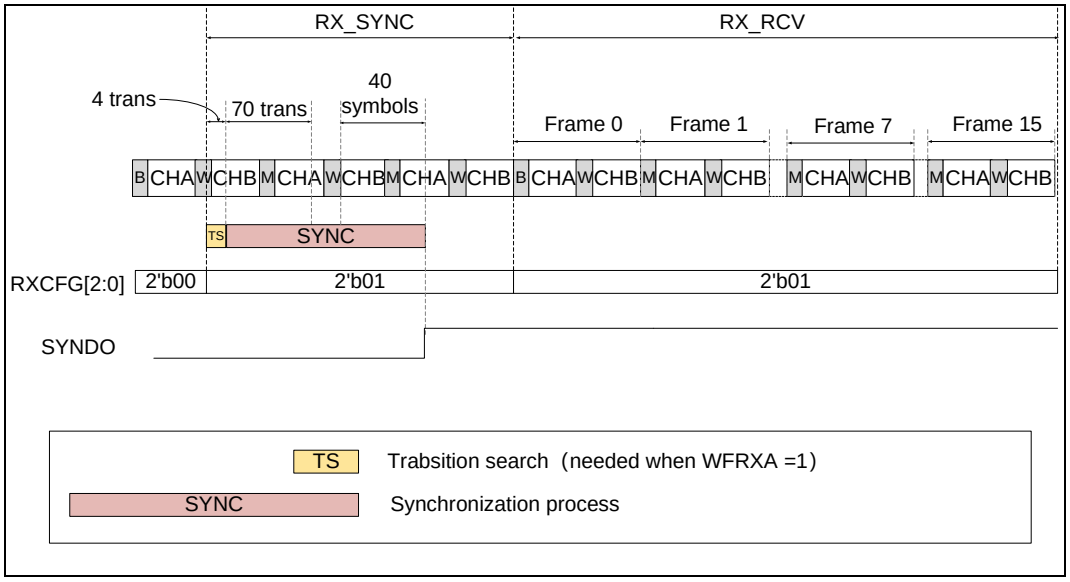



**Figure 33-8. Synchronization process scheduling**




### 33.3.5. RSPDIF state machine

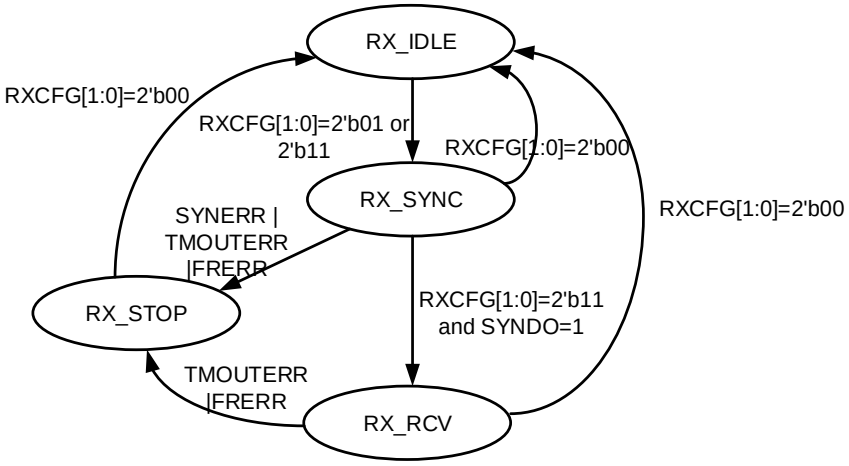
The RSPDIF state machine includes the following states, as shown in ***Table 33-8. RSPDIF state***.

**Table 33-8. RSPDIF state**

| RSPDIF state | Description |
|---|---|
| RX_IDLE | In idle state, RSPDIF is disabled, rspdif_ckdomain is reset, and rspdif_pclk domain function is normal. |
| RX_SYNC | In synchronization status, RSPDIF is synchronized with the data stream, the threshold is updated regularly, and the user data and channel status can be obtained through interrupt or DMA. |
| RX_RCV | In receiving status: RSPDIF is synchronized with the data stream, and the threshold is updated regularly. Users, channel status and audio data can be obtained through interrupts or DMA. When RXCFG[1:0] turns to 2’b11, RSPDIF waits for the "B" preamble before starting to save audio data. |
| RX_STOP | In stop status: RSPDIF is no longer synchronized, user and channel status and audio data reception stop. |

The ***Figure 33-9. RSPDIF states*** shows how to switch between RSPDIF states.

**Figure 33-9. RSPDIF states**




### RX_IDLE

The state machine will enter the RX_IDLE state from the following conditions:

- Set RXCFG[1:0] to 2'b00.
- When in RX_SYNCstate, set RXCFG[1:0] to 2'b00.
- When in RX_RCV state, set RXCFG[1:0] to 2'b00.
- When in RX_STOP state, set RXCFG[1:0] to 2'b00.

The next state of RX_IDLE is:

- RX_SYNC: Set RXCFG[1:0] to 2'b01 or 2'b11.

**Note:** The software can set RXCFG[1:0] to 0 at any time, and RSPDIF immediately returns to the RX_IDLE state. If the DMA transfer is in progress, it will wait for the DMA transfer completed.

### RX_SYNC

The state machine will enter the RX_SYNC state from the following conditions:

- When in RX_IDLE state, set RXCFG[1:0] to 2'b01 or 2'b11.

The next state of RX_SYNC is:

- RX_IDLE : Set RXCFG[1:0] to 2'b00.
- RX_RCV : Set RXCFG[1:0] to 2'b11 and synchronization is completed successfully (SYNDO = 1).
- RX_STOP : Synchronization fails (If the number of synchronization retries has been set, the maximum value has been reached) or the received data has not been decoded correctly (FRERR or SYNERR or TMOUTERR = 1).

**Note:** When the synchronization is completed, if RXCFG[1:0] = 01, the RSPDIF is still in this state.

### RX_RCV

The state machine will enter the RX_RCV state from the following conditions:

- When in RX_SYNC state, s et RXCFG[1:0] to 2'b11 and synchronization is completed successfully (SYNDO = 1).

The next state of RX_RCV is:

- RX_IDLE : Set RXCFG[1:0] to 2'b00.
- RX_STOP：The received data has not been decoded correctly (FRERR or TMOUTERR = 1).

### RX_STOP

The state machine will enter the RX_STOP state from the following conditions:

- When in RX_SYNC state, synchronization fails (If the number of synchronization retries has been set, the maximum value has been reached) or the received data has not been decoded correctly (FRERR or SYNERR or TMOUTERR = 1).
- When in RX_RCVstate, the received data has not been decoded correctly (FRERR or TMOUTERR = 1).

The next state of RX_STOP is:

- RX_IDLE：Set RXCFG[1:0] to 2'b00.

**Note:** When RXCFG[1:0] is set to 0, the RSPDIF is disabled, meaning that all the state machines are reset, and RX BUFFER is flushed, as well that flags FRERR, SYNERR and TMOUTERR are reset.

### RSPDIF_CTL register access permissions in different states

In the different state stages of the RSPDIF state machine, the access rights to different bit fields of the RSPDIF_CTL register are different. Please refer to ***Table 33-9. RSPDIF_CTL register access permissions in different states*** for details. SPDIF_CTL bit access characteristics in different RSPDIF states. The hardware handles this way to avoid incorrect configuration of the RSPDIF_CTL register. Please note that even user can modify the PTNCPEN, CUNCPEN, VNCPEN and PNCPEN bits in any state of RSPDIF, but these modifications will not affect the value already saved in the DATA register.

**Table 33-9. RSPDIF_CTL register access permissions in different states**

| RSPDIF state | RXCHSEL[2:0] | WFRXA | MAXRT[1:0] | CFCHSEL | DMACBEN | PTNCPEN | CUNCPEN | VNCPEN | PNCPEN | RXDF[1:0] | RXSTEOMEN | DMAREN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RX_IDLE (RXCFG[1:0] = 2'b00) | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| RSPDIF state | RXCHSEL[2:0] | WFRXA | MAXRT[1:0] | CFCHSEL | DMACBEN | PTNCPEN | CUNCPEN | VNCPEN | PNCPEN | RXDF[1:0] | RXSTEOMEN | DMAREN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RX_SYNC (RXCFG[1:0] = 2'b01) | r | r | r | r | rw | rw | rw | rw | rw | rw | rw | rw |
| RX_RCV (RXCFG[1:0] = 2'b11) | r | r | r | r | rw | rw | rw | rw | rw | r | r | rw |

## 33.3.6. RSPDIF data reception management

### RSPDIF double data buffer

RSPDIF implements the double buffering function when data is received. The receiving data double buffer is composed of a 32-bit RX buffer and RSPDIF_DATA register. When the RSPDIF_DATA register is empty and the conversion between the parity bit (P) and the next preamble has been detected, the data in the RX buffer will be immediately transferred to the RSPDIF_DATA register.

The double buffer mechanism of data improves the tolerance of delay. The maximum delay allowed is $T_{SAMPLE}$ - $2T_{pclk}$ - $2T_{rspdif_ck}$ ($T_{SAMPLE}$ is the audio sampling rate of the received stereo audio samples, $T_{PCLK}$ is the period of the rspdif_pclk clock, and $T_{rspdif_ck}$ is the period of rspdif_ck clock.).

The software can obtain the received data by reading the RSPDIF_DATA register or DMA. Recommended DMA operation, please refer to ***DMA function*** for details.

### RSPDIF_DATA register

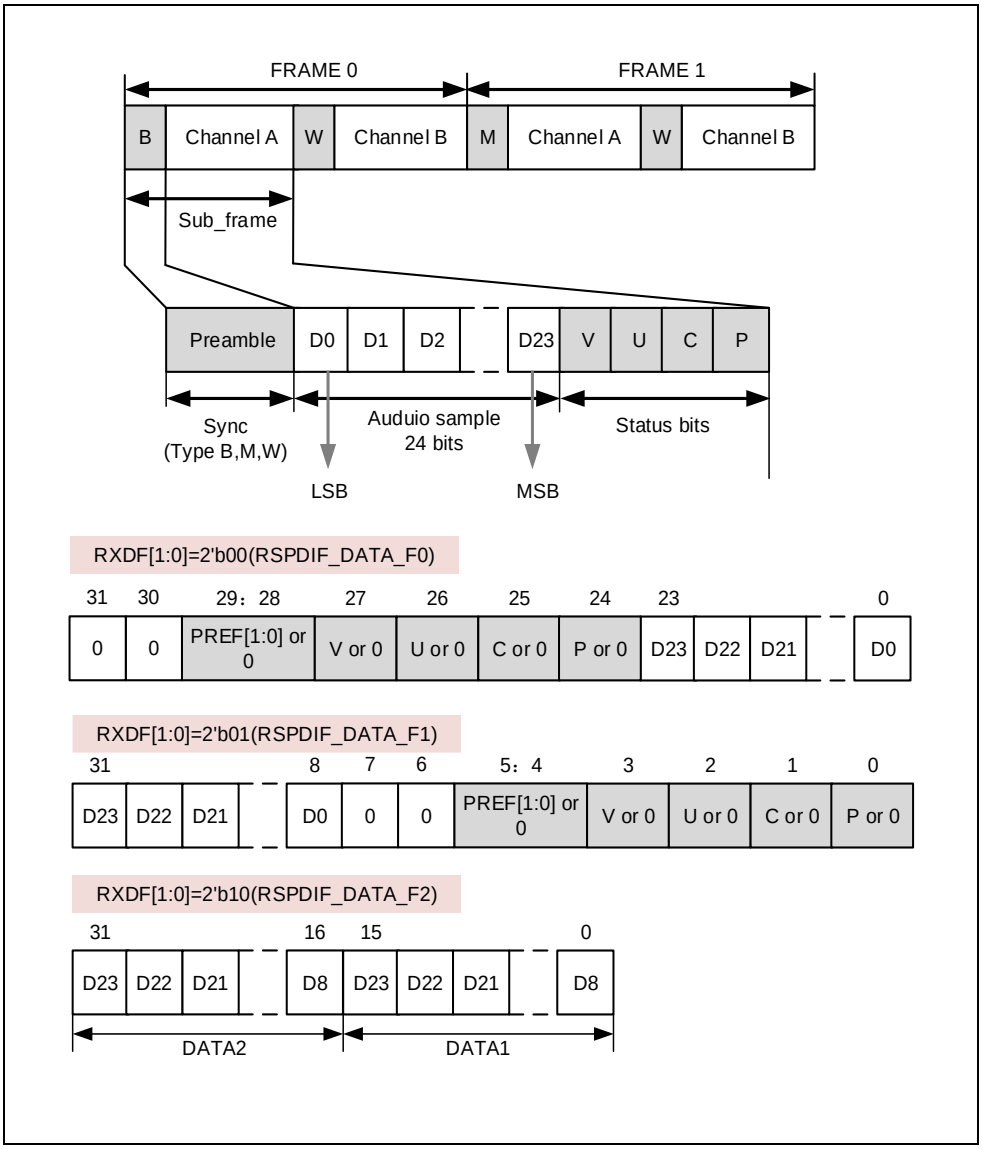
Except for V, U, C, P status bits and headers, each subframe includes up to 24 bits of data. The RSPDIF_DATA register has three formats according to the value of RXDF[1:0] in the RSPDIF_CTL register, and there are three processing methods for the received audio data stream:

- When RXDF[1:0] = 2'b00, the format of the data register is the format described by RSPDIF_DATA_F0, and the data is forced to be right-aligned. The PREF\C\U\V\P bit can be enabled or forced to 0 as required.
- When RXDF[1:0] =2'b01, the format of the data register is the format described by RSPDIF_DATA_F1, and the data is forced to be left-aligned. The PREF\C\U\V\P bit can be enabled or forced to 0 as required.
- When RXDF[1:0] = 2'b11, the format of the data register is the format described by RSPDIF_DATA_F2. This format uses only 16 bits per subframe in non-linear mode. The data of two consecutive sub-frames will be stored in a RSPDIF_DATA register. The format PREF\C\U\V\P bits and data bits cannot be mixed, but the software can still be obtained by reading the RSPDIF_CHSTAT register or a dedicated DMA channel. When

RXSTEOMEN = 1, there is no risk of misalignment (that is, the data from channel A is always stored in RSPDIF_DATA[31:16]). If RXSTEOMEN = 0, there is a risk of misalignment in case of excess. In this case, RSPDIF_DATA[31:16] always contains the oldest value, and RSPDIF_DATA[15:0] always contains the most recent value (***Figure 33-10. RSPDIF_DATA register format***. ).

**Note:** This document describes three data registers: RSPDIF_DATA_Fx (x = 0,1,2), but there is actually only one physical data register.

**Figure 33-10. RSPDIF_DATA register format**

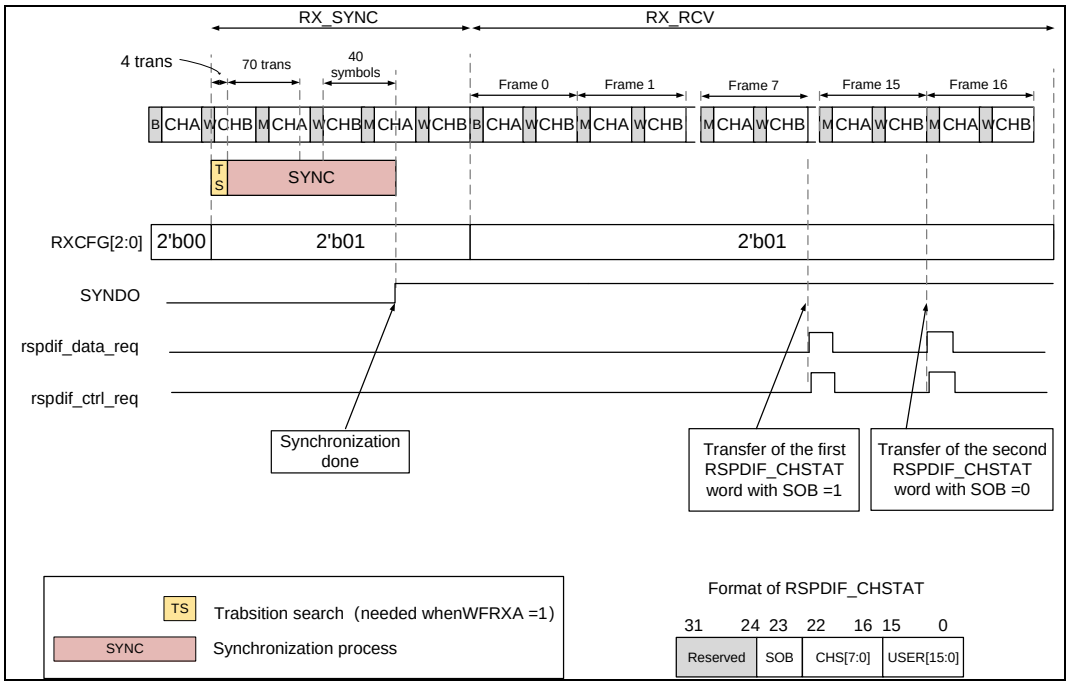



## Dedicated control flow

The RSPDIF provides a dedicated channel for receiving user data U and channel status C. By configuring the CFCHSEL bit in the RSPDIF_CTL register, user can select to obtain the channel status bit C from channel A or channel B. When RXCFG[1:0] is set to 2'b01 or 2'b11, the acquisition will start after the synchronization phase ends. When 8 channel status bits and

16 user data bits are received, pack them and store them in the RSPDIF_CHSTAT register. If the DMACBEN bit of the RSPDIF_CTL register is configured as 1, and the DMA function of the control flow is enabled, a DMA request is triggered. If CHS[0] is the first status bit of the new block, the SOB bit is set to 1. Please refer to ***RX Channel status register (RSPDIF_CHSTAT)***.

**Figure 33-11. Channel/user data format**




## Mixing data and control flow

When RXDF[1:0] = 2'b00 or 2'b01, the RSPDIF_DATA register format is RSPDIF_DATA_F0 or RSPDIF_DATA_F1, which allows the V, U, C, P status bits, preamble and data mixed reception management. By configuring the RSPDIF_CTL register, the user can choose which contents of the V, U, C, P status bits, and preamble are stored in the RSPDIF_DATA register flexibly, as shown in ***Table 33-10. Mixing data and control flow***.

**Table 33-10. Mixing data and control flow**

| BIT | value | Description |
|---|---|---|
| PNCPEN | 1 | Parity error information is masked |
| | 0 | Parity error information is copied into RSPDIF_DATA |
| VNCPEN | 1 | Validity information is masked |
| | 0 | Validity information is copied into RSPDIF_DATA |
| CUNCPEN | 1 | Channel status, and user data information are masked |
| | 0 | Channel status, and user data information are copied into RSPDIF_DATA |
| PTNCPEN | 1 | Preamble type is masked |
| | 0 | Preamble type is copied into RSPDIF_DATA |

### 33.3.7. RSPDIF clock management

From the RSPDIF block diagram shown in ***Figure 33-1. RSPDIF block diagram***, the RSPDIF block needs the rspdif_pclk clock for the registers interface and the rspdif_ck clock for the RSPDIF DEC module. rspdif_ck should not be phase locked. In the SYNC block, all signals passing through these clock domains are resynchronized.

In order to decode the incoming S/PDIF stream correctly, the decoding clock rspdif_ck should be at least 11 times higher than the maximum symbol rate or 704 times higher than the audio sampling rate to resample the received data. For example, if the audio rate is 192KHz, and the symbol rate is 12.288MHz (audio rate * number of channels * number of sub-frames), at least 135.2MHz decoding clock (symbol rate * 11) is required. For rspdif_pclk, rspdif_pclk cannot be less than the symbol rate.

**Table 33-11. Minimum rspdif_ck frequency and audio sampling rate**

| Symbol Rate | Minimum rspdif_ck frequency | Audio |
|---|---|---|
| 3.072 MHz | 33.8 MHz | For 48 kHz stream |
| 6.144 MHz | 67.6 MHz | For 96 kHz stream |
| 12.288 MHz | 135.2 MHz | For 192 kHz stream |

In RSPDIF module,

RSPDIF_DEC: RSPDIF_DEC is designed to decode the input RSPDIF bits stream. When RSPDIF is not enable, the clock of RSPDIF_DEC module is gated. The source clock of this module is rspdif_clk.

#### RSPDIF symbol clock

When RSPDIF is decoding input audio stream, a symbol clock rspdif_symbol_ck is constructed using WIDTH24, WIDTH40 and the value of the symbol boundary. It can be used as a reference core clock for other audio devices (such as SAI or I2S), and can be used for the RSPDIF to I2S bridging function.

When receiving the subframe synchronization preamble, the falling and rising edges of the symbol clock are constructed from the WIDTH24 and WIDTH40 values. When RSPDIF is RX_STOP or RX_IDLE, WIDTH24 and WIDTH40 can also be used to generate the symbol clock. When receiving a subframe, RSPDIF uses symbol boundaries to generate rising edges, use WIDTH24 and WIDTH40 generate falling edges, and the symbol clock's duty cycle is close to 50%.

However, when RSPDIF switches from using WIDTH24 and WIDTH40 to generate a symbol clock to generating a symbol clock from a symbol boundary, the symbol clock duty cycle can be changed, and vice versa. When the symbol clock generation mode is switched or when the S/PDIF signal is re-sampled using the rspdif_ck clock, the symbol clock may be greatly jittered.

Configure the SCKEN bit in the RSPDIF_CTL register to enable or disable the generation of the symbol clock. Configure the BKSCKEN bit in the RSPDIF_CTL register to enable or

disable the generation of the backup symbol clock. However, when the flag SYNERR is set to ‘1’, neither the symbol clock nor the backup clock can be generated, since there is no synchronization. When both SCKEN and BKSCKEN are set to ‘1’, the symbol clock will loose some transitions when the RSPDIF switches from RX_SYNC or RX_RCV to RX_STOP, or RX_IDLE.

For the specific generation conditions of the symbol clock, please refer to ***Table 33-12. Conditions of RSPDIF_symbol_ck generation***.

**Table 33-12. Conditions of RSPDIF_symbol_ck generation**

| RSPDIF | SCKEN | BKSCKEN | valid data are received via the selected RSPDIF CH | valid values for WIDTH40 and WIDTH24 | WIDTH40 and WIDTH24 contain the valid values from the previous synchronization | no transitions detected on the selected RSPDIF input | complete the Synchronization (SYNDO = '1') | RSPDIF_symb_ck state |
|---|---|---|---|---|---|---|---|---|
| Any state | 0 | x | x | x | x | x | x | Disable |
| RX_IDLE | 1 | 0 | x | x | x | x | 1 | No output |
| | | 1 | x | 0 | x | x | x | |
| | 1 | 1 | x | 1 | x | x | x | output |
| RX_SYNC | 1 | 0 | x | x | x | x | 0 | No output |
| | | 1 | x | 0 | x | x | 0 | |
| | 1 | 0 | x | x | x | x | 1 | output |
| | | 1 | x | x | x | x | 1 | |
| | | | x | x | 1 | x | 0 | |
| STATE_REV | 1 | 0 | x | x | x | 1 | x | No output |
| | | 1 | x | x | x | 1 | x | |
| | 1 | 0 | 1 | x | x | x | x | output |
| | | 1 | 1 | x | x | x | x | |
| RX_STOP | 1 | 0 | x | x | x | x | x | No output |
| | | 1 | x | 0 | x | x | x | |
| | 1 | 1 | x | 1 | x | x | x | output |

**Note:** "0" means that the condition must not be met; "1" means that the condition must be met; "x" means that the condition has no effect on the generation of the symbol clock.

### 33.3.8. DMA function

RSPDIF can obtain data and control information through the corresponding dedicated DMA channel.. Please refer to the ***Direct memory access controller (DMA)*** for specific channels used. Set the DMAREN bit in the RSPDIF_CTL register can enable the RSPDIF block data DMA transfer function. When RSPDIF_DATA is not empty, a data transfer request will be sent to the DMA, and the DMA will read the data directly. Set the DMACBEN bit in the RSPDIF_CTL register can enable the RSPDIF block DMA function of channel and user

information. For DMA usage of control data, please refer to ***Table 33-9. RSPDIF_CTL register access permissions in different states***.

It is not recommended to configure the DMAREN or DMACBEN bit to enable DMA transfer after beginning data reception.

### 33.3.9. Status、error and interrupt

#### Status flags

**Reception events for data flow (RBNE)**

When the RX buffer is not empty, the RBNE bit is set. Indicates that a data is received and when the RSPDIF_DATA register is empty and the conversion between the parity bit (P) and the next preamble has been detected, the data in the RX buffer will be immediately transferred to the RSPDIF_DATA register. The software can obtain this data by reading RSPDIF_DATA register.

**Reception event for control flow (CBNE)**

When the RSPDIF_CHSTAT is not empty, the CBNE bit is set. Indicates that the control data is received and stored in the RSPDIF_CHSTAT at this time. The software can obtain this data by reading RSPDIF_CHSTAT.

**Synchronization done (SYNDO)**

When RSPDIF can measure the duration of 24 and 40 consecutive symbols correctly and SYNC is completed, this bit is set. Indicates that the synchronization is completed.

**Start of new block interrupt (SYNDB)**

When preamble B has been detected, this bit is set. Represented as the beginning of a block.

**Note:** SYNDB event can only occur when RSPDIF is synchronized to the input stream (SYNDO = 1).

#### Error flags

**Frame error (FERR)**

When the conversion sequence is incorrect in the 28 information bits or the preamble appears in an unexpected position or the expected preamble is not received, RSPDIF frame error occurs and the FRERR bit is set. If the RXDCERRIE bit of the RSPDIF_INTEN register is set, a corresponding interrupt will be generated. Set RXCFG[1:0] to 2’b00 to clear the error flag.

**Synchronization error (SYNERR)**

When the synchronization fails (if the maximum retries MAXTR is set, MAXTR has been

exceeded), RSPDIF synchronization error occurs, and the SYNERR bit is set. If the RXDCERRIE bit of the RSPDIF_INTEN register is set, a corresponding interrupt will be generated. Set RXCFG[1:0] to 2'b00 to clear the error flag.

**Timeout error (TMOUTERR)**

When no conversion is detected during the 8192 cycle of the rspdif_ck clock and TWCNT overflows, RSPDIF timeout error occurs, and the TMOUTERR bit is set. If the RXDCERRIE bit of the RSPDIF_INTEN register is set, a corresponding interrupt will be generated. Set RXCFG[1:0] to 2'b00 to clear the error flag.

**Parity error (PERR)**

When the number of 0 or 1 is not an even number among the 28 information bits in a subframe, RSPDIF parity error occurs, and the PERR bit is set. If the PERRIE bit in the SPDIF_INTEN register is set, a corresponding interrupt will be generated. Set the PERRC bit to 1, to clear the PERR flag.

**Note:** Even if the interrupt is suspended, receiving incoming data will not be suspended, and RSPDIF will continue to send data to RSPDIF_DATA. If the software wants to ensure the consistency between the data read in the RSPDIF_DATA register and the value of the PERR bit, the PNCPEN bit must be set to 0.

**Overrun error (RXORERR)**

When both RSPDIF_DATA and RX buffer are full, while RSPDIF_DEC still writes a new data into the RX buffer, RSPDIF overrun error occurs, RXORERR is set, and this new data will not be written into the buffer. If the RXORERRIE bit of the RSPDIF_INTEN register is set, a corresponding interrupt will be generated. Set the RXORERRC bit to 1 to clear the RXORERR flag.

**Note:** Even if the RXORERR flag is suspended, when RXSTEOMEN = 0 and the RX buffer is empty, the next incoming data will still be stored. See ***Figure 33-12. RSPDIF overrun error when RXSTEOMEN = 0 and RXDF[1:0] = 2'b0x*** and ***Figure 33-13. RSPDIF overrun error when RXSTEOMEN = 0 and RXDF[1:0] = 2'b10***.

**Figure 33-12. RSPDIF overrun error when RXSTEOMEN = 0 and RXDF[1:0] = 2'b0x**

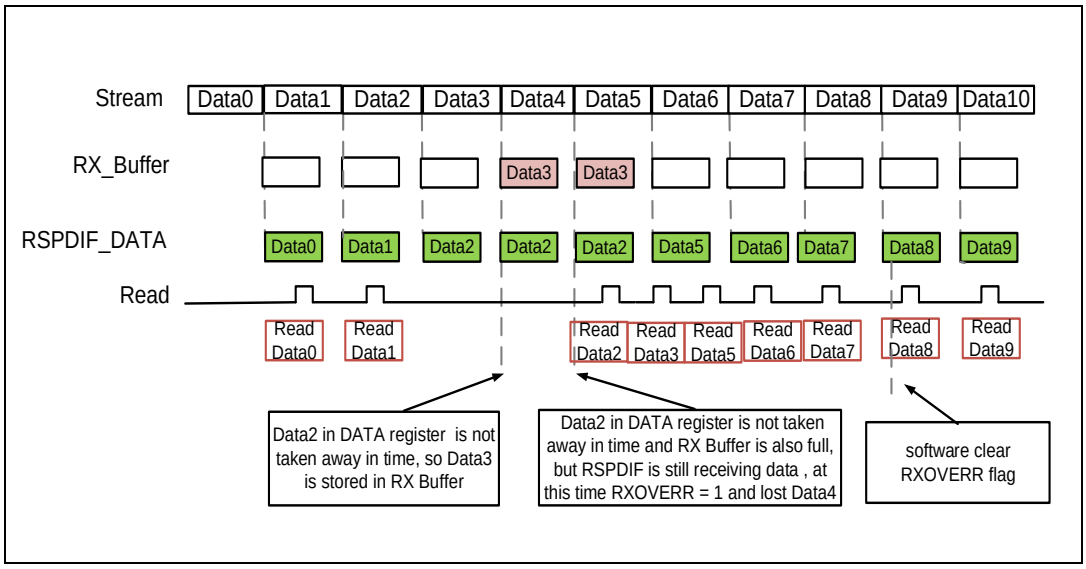
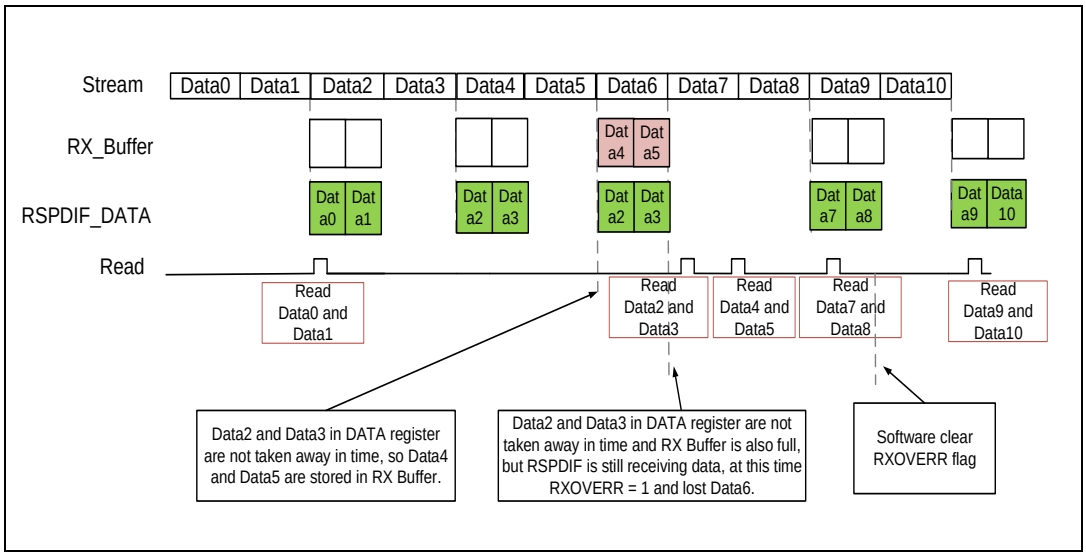



**Figure 33-13. RSPDIF overrun error when RXSTEOMEN = 0 and RXDF[1:0] = 2'b10**




If RXSTEOMEN = 1, RSPDIF transmits stereo data. In order to avoid the problem of misalignment of the left and right channels, RSPDIF must discard the second data, even if there is space in the RX buffer. The incoming data can be written to the RX buffer normally, even if the RXORERR flag is still suspended. Refer to ***Figure 33-14. RSPDIF overrun error when RXSTEOMEN = 1 and RXDF[1:0] = 2'b0x*** and ***Figure 33-15. RSPDIF overrun error when RXSTEOMEN = 1 and RXDF[1:0] = 2'b10***.

**Figure 33-14. RSPDIF overrun error when RXSTEOMEN = 1 and RXDF[1:0] = 2'b0x**

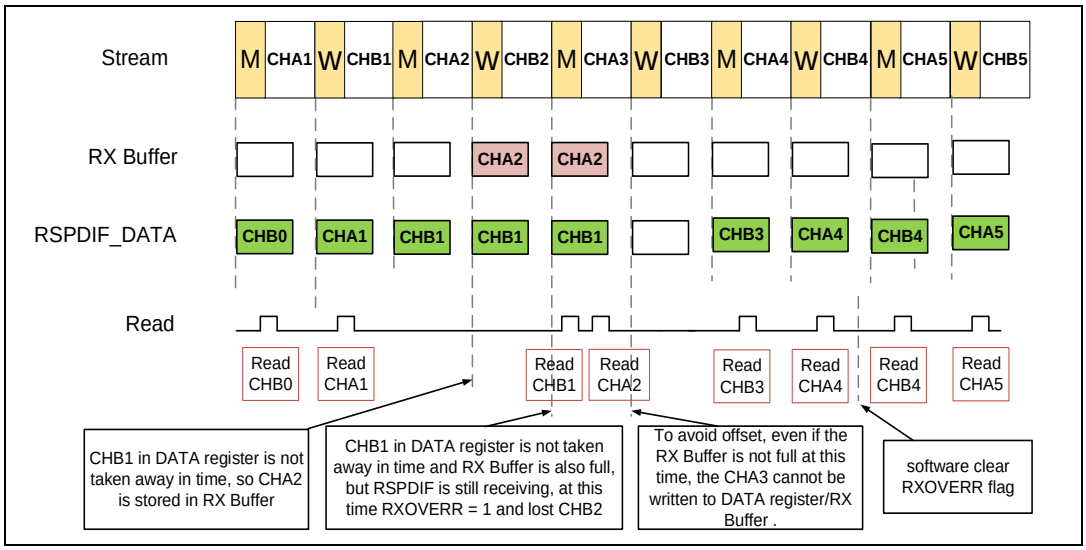
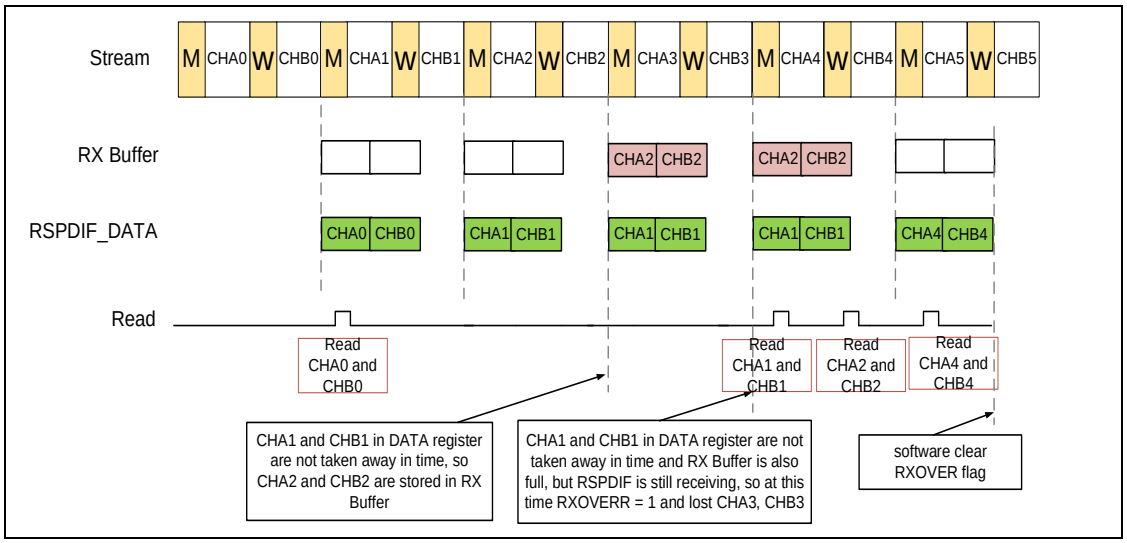

Stream
M CHA1 W CHB1 M CHA2 W CHB2 M CHA3 W CHB3 M CHA4 W CHB4 M CHA5 W CHB5
RX Buffer
CHA2
CHA2
RSPDIF_DATA
CHB0
CHA1
CHB1
CHB1
CHB1
CHB3
CHA4
CHB4
CHA5
Read
Read CHB0
Read CHA1
Read CHB1
Read CHA2
Read CHB3
Read CHA4
Read CHB4
Read CHA5
CHB1 in DATA register is not taken away in time, so CHA2 is stored in RX Buffer
CHB1 in DATA register is not taken away in time and RX Buffer is also full, but RSPDIF is still receiving, at this time RXOVERR = 1 and lost CHB2
To avoid offset, even if the RX Buffer is not full at this time, the CHA3 cannot be written to DATA register/RX Buffer .
software clear RXOVERR flag

**Figure 33-15. RSPDIF overrun error when RXSTEOMEN = 1 and RXDF[1:0] = 2'b10**




## Interrupts

The RSPDIF interrupt events and flags are listed in ***Table 33-13. RSPDIF interrupt events***.

**Table 33-13. RSPDIF interrupt events**

| Flag | Interrupt event | Clear method | Enable control bit |
|---|---|---|---|
| SYNDO | SYNC is completed | Set SYNDBC bit in RSPDIF_STATC register to 1 | SYNDOIE |
| RBNE | RSPDIF receive buffer is not empty | Read RSPDIF_DATA register | RBNEIE |
| PERR | In the 28 information bits of one subframe, 0 and 1 are not even numbers | Set PERRC bit in RSPDIF_STATC register to 1 | PERRIE |
| RXORERR | Both RSPDIF_DATA and RX buffer are full, and RSPDIF_DEC still write a new data to RX buffer | Set RXORERRC bit in RSPDIF_STATC register to 1 | RXORERRIE |
| CBNE | RSPDIF control flow receive buffer is not empty | Read RSPDIF_CHSTAT register | CBNEIE |
| SYNDB | Preamble B has been detected | Set SYDBC bit in RSPDIF_STATC register to 1 | SYNDBIE |
| SYNERR | The synchronization fails | Clear RXCFG bits in RSPDIF_CTL register | RXDCERRIE |
| FRERR | The conversion sequence is incorrect in the 28 information bits or the preamble appears in an unexpected position or the expected preamble is not received | | |
| TMOUTERR | TWCNT overflows | | |

## 33.4. Register definition

RSPDIF base address: 0x4000 4000

### 33.4.1. Control register (RSPDIF_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | BKSCKEN | SCKEN | Reserved | RXCHSEL[2:0] | | |
| | | | | | | | | | | rw | rw | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | WFRXA | MAXRT[1:0] | | CFCHSEL | DMACBEN | PTNCPEN | CUNCPEN | VNCPEN | PNCPEN | RXDF[1:0] | | RXSTEOMEN | DMAREN | RXCFG[1:0] | |
| | rw | rw | | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21 | BKSCKEN | Backup symbol clock enable<br>This bit is set and cleared by software.<br>0: The RSPDIF does not generate a backup symbol clock.<br>1: The RSPDIF generates a backup symbol clock if SCKEN = 1<br>Please refer to ***RSPDIF clock management*** for information on how to program this field. |
| 20 | SCKEN | Symbol clock enable<br>This bit is set and cleared by software.<br>0: The RSPDIF does not generate a symbol clock<br>1: The RSPDIF generates a symbol clock |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | RXCHSEL[2:0] | RSPDIF input channel selection<br>000: RSPDIF_CH0 selected<br>001: RSPDIF_CH1 selected<br>010: RSPDIF_CH2 selected<br>011: RSPDIF_CH3 selected<br>100 ~ 111: reserved |
| 15 | Reserved | Must be kept at reset value. |
| 14 | WFRXA | Wait for the four valid transition signal of the selected RSPDIF channel<br>This bit is set and cleared by software. |

| | | |
|---|---|---|
| | | 0: Before starting the synchronization process, RSPDIF does not wait for the valid conversion signal from the selected RSPDIF channel.<br>1: Before starting the synchronization process, RSPDIF wait for the valid conversion signal from the selected RSPDIF channel. |
| 13:12 | MAXRT[1:0] | Maximum number of retries allowed during the RSPDIF synchronization phase<br>00: Do not allow retry (only one chance)<br>01: Allow up to 3 retries<br>10: Allow up to 15 retries<br>11: Allow up to 63 retries |
| 11 | CFCHSEL | The control flow acquires channel state source selection<br>This bit is set and cleared by software.<br>0: Gets channel status from channel A<br>1: Gets channel status from channel B |
| 10 | DMACBEN | Control buffer DMA enable for control flow<br>This bit is set and cleared by software.<br>0: Disable control flow DMA mode.<br>1: Enable control flow DMA mode.<br>When this bit is set, a DMA request is generated whenever the CBNE flag is set. |
| 9 | PTNCPEN | Preamble type bits no copy enable bit<br>This bit is set and cleared by software.<br>0: Copy the preamble type bits into the RSPDIF_DATA<br>1: Do not copy the preamble type bits into the RSPDIF_DATA, but write 0 instead. |
| 8 | CUNCPEN | Channel status and user bits no copy enable bit<br>This bit is set and cleared by software.<br>0: Copy the channel status and user bits into the RSPDIF_DATA.<br>1: Do not copy the channel status and user bits into the RSPDIF_DATA, but write 0 instead. |
| 7 | VNCPEN | Validity bit no copy enable bit<br>This bit is set and cleared by software.<br>0: Copy the validity bit into the RSPDIF_DATA.<br>1: Do not copy the validity bit into the RSPDIF_DATA, but write 0 instead. |
| 6 | PNCPEN | Parity error bit no copy enable bit<br>This bit is set and cleared by software.<br>0: Copy the parity error bit into the RSPDIF_DATA.<br>1: Do not copy the parity error bit into the RSPDIF_DATA, but write 0 instead. |
| 5:4 | RXDF[1:0] | RX data format selection<br>This bit is set and cleared by software.<br>00: The data format is described in the RSPDIF_DATA_F0 register, audio data is right aligned (LSB)<br>01: The data format is described in the RSPDIF_DATA_F1 register, audio data is |

| | | |
|---|---|---|
| | | left aligned (MSB)<br>10: The data format is described in the RSPDIF_DATA_F2 register, which packs two 16-bit audio data into one 32-bit data<br>11: reserved |
| 3 | RXSTEOMEN | RX stereo mode enable<br>This bit is set and cleared by software.<br>0: MONO mode<br>1: STEREO mode<br>This bit is used in case of overrun situation in order to handle misalignment |
| 2 | DMAREN | Receiver DMA enable for data flow<br>This bit is set and cleared by software.<br>0: Disable reception DMA mode.<br>1: Enable reception DMA mode.<br>When this bit is set, the DMA request is generated whenever the RBNE flag is set. |
| 1:0 | RXCFG[1:0] | RSPDIF configuration<br>00: Disable RSPDIF<br>01: Enable RSPDIF synchronization only<br>10: Reserved<br>11: Enable RSPDIF |

## 33.4.2. Interrupt enable register (RSPDIF_INTEN)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | RXDCERRIE | SYNDOIE | SYNDBIE | RXORERRIE | PERRIE | CBNEIE | RBNEIE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | RXDCERRIE | RSPDIF data decoding error interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable<br>When this bit is set, a RSPDIF data decoding error interrupt is generated if SYNERR, TMOUTERR or FRERR bit in the RSPDIF_STAT register is set. |

| | | |
|---|---|---|
| 5 | SYNDOIE | Synchronization done interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable<br>When this bit is set, a synchronization done interrupt is generated if SYNDO bit in the RSPDIF_STAT register is set. |
| 4 | SYNDBIE | Synchronization block detected interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable<br>When this bit is set, a synchronization block detected interrupt is generated if SYNDB in the RSPDIF_STAT register is set . |
| 3 | RXORERRIE | RX overrun error interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable<br>When this bit is set, a RX overrun error interrupt is generated if RXORERR in the RSPDIF_STAT register is set. |
| 2 | PERRIE | Parity error interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable<br>When this bit is set, a parity error interrupt is generated if PERR in the RSPDIF_STAT register is set. |
| 1 | CBNEIE | RSPDIF_CHSTAT register no empty interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable<br>When this bit is set, a RSPDIF_CHSTAT register no-empty interrupt is generated if CBNE in the RSPDIF_STAT register is set. |
| 0 | RBNEIE | RSPDIF_DATA register no empty interrupt enable<br>This bit is set and cleared by software.<br>0: Disable<br>1: Enable.<br>When this bit is set, a RSPDIF_DATA register no-empty interrupt is generated if RBNE in the RSPDIF_STAT register is set. |

### 33.4.3. Status register (RSPDIF_STAT)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | CKCNT5[14:0] | | | | | | | | |
| | | | | | | | | r | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | TMOUTERR | SYNERR | FRERR | SYNDO | SYNDB | RXORERR | PERR | CBNE | RBNE |
| | | | | | | | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:16 | CKCNT5[14:0] | The number of consecutive time clock cycles of 5 symbols counted using rspdif_ck<br>This value can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of rspdif_ck.<br>For example if the rspdif_ck is fixed to 84 MHz, and CKCNT5 = 147d. The estimated sampling rate of the S/PDIF stream is:<br>Fs = 5 x rspdif_ck / (CKCNT5 x 64) ~ 44.6 kHz, so the closest standard sampling rate is 44.1 kHz.<br>Note that CKCNT5 is updated by the hardware when SYNDO goes high, and then every frame. |
| 15:9 | Reserved | Must be kept at reset value. |
| 8 | TMOUTERR | Time out error<br>0: No timeout error is detected<br>1: Timeout error is detected<br>When the counter TWCNT value reaches its maximum value, the bit is set to 1 by the hardware. It indicates that the time between transitions is too long. This usually means that there is no valid signal on the RSPDIF_CH input.<br>If RXDCERRIE=1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Set RXCFG[1:0] to 2'b00 in RSPDIF_CTL register to clear this bit. |
| 7 | SYNERR | Synchronization error<br>0: No synchronization error is detected<br>1: Synchronization error is detected<br>When synchronization fails due to the number of MAXRT retries, the bit is set to 1 by the hardware.<br>If RXDCERRIE=1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Set RXCFG[1:0] to 2'b00 in RSPDIF_CTL register to clear this bit. |
| 6 | FRERR | Frame error<br>0: no Manchester coding error detected<br>1: Manchester coding error detected<br>When an error occurs during receiving data, the bit is set to 1 by the hardware. Preamble does not appear in expected position, short transitions are not grouped by pair. This bit can be set to 1 by the hardware only when synchronization is |

| | | |
|---|---|---|
| | | complete (SYNDO = 1).<br>If RXDCERRIE=1 in register RSPDIF_INTEN, an interrupt is generated.<br>Set RXCFG[1:0] to 2'b00 in RSPDIF_CTL register to clear this bit. |
| 5 | SYNDO | Synchronization done<br>0: Synchronization is pending<br>1: Synchronization is completed<br>When the synchronization phase is properly completed, the bit is set to 1 by the hardware..<br>If SYNDOIE = 1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Set SYNDOC bit in RSPDIF_STATC register to clear this bit. |
| 4 | SYNDB | Synchronization block detected<br>0: No "B" preamble detected<br>1: "B" preamble has been detected<br>When a "B" preamble has been detected, the bit is set to 1 by the hardware.<br>If SYNDBIE = 1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Set SYNDBC bit in RSPDIF_STATC register to clear this bit. |
| 3 | RXORERR | RX overrun error<br>0: No Overrun error<br>1: Overrun error is detected<br>When RSPDIF is still receiving data while RBNE = 1 and both RSPDIF_DATA and RX buffer are full, the bit is set to 1 by the hardware.<br>If RXORERRIE=1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Set RXORERRC bit in RSPDIF_STATC register to clear this bit.<br>**Note:** When this bit is set, the RSPDIF_DATA register content is not lost but the last data received are lost. |
| 2 | PERR | Parity error<br>0: No parity error<br>1: Parity error<br>When the received subframe contains an odd number of 0 and 1 in the data and status bits, the bit is set to 1 by the hardware.<br>If PERRIE = 1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Set PERRC bit in RSPDIF_STATC register to clear this bit. |
| 1 | CBNE | RSPDIF_CHSTAT register is not empty<br>0: No control word available on RSPDIF_CHSTAT register<br>1: A control word is available on RSPDIF_CHSTAT register<br>When there is control flow data in the RSPDIF_CHSTAT register, the bit is set to 1 by the hardware.<br>If CBNEIE = 1 in the RSPDIF_INTEN register, an interrupt is generated.<br>Read RSPDIF_CHSTAT register to clear this bit. |
| 0 | RBNE | RX buffer is not empty<br>0: Data is not received |

1: Received data is ready to be read.

When a valid data is available into RX buffer, the bit is set to 1 by the hardware.

If RBNEIE = 1 in the RSPDIF_INTEN register, an interrupt is generated.

Read RSPDIF_DATA register to clear this bit.

### 33.4.4. Status flag clear register (RSPDIF_STATC)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | SYNDOC | SYNDBC | RXORERRC | PERRC | Reserved | |
| | | | | | | | | | | w | w | w | w | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5 | SYNDOC | clears the synchronization done flag<br>Set this bit to clear the SYNDO flag in the RSPDIF_STAT register. |
| 4 | SYNDBC | clears the synchronization block detected flag<br>Set this bit to clear the SYNDB flag in the RSPDIF_STAT register. |
| 3 | RXORERRC | clears the RX overrun error flag<br>Set this bit to clear the OVR flag in the RSPDIF_STAT register. |
| 2 | PERRC | clears the parity error flag<br>Set this bit to clear the PERR flag in the RSPDIF_STAT register. |
| 1:0 | Reserved | Must be kept at reset value. |

### 33.4.5. RX data register (RSPDIF_DATA)

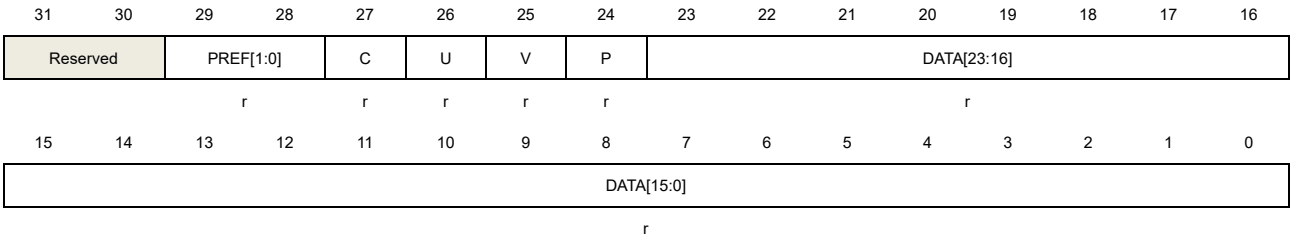
Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

According to the value of RXDF[1:0], the receive data register has three different formats. When RXDF[1:0]=2'b00, the format is as follows:

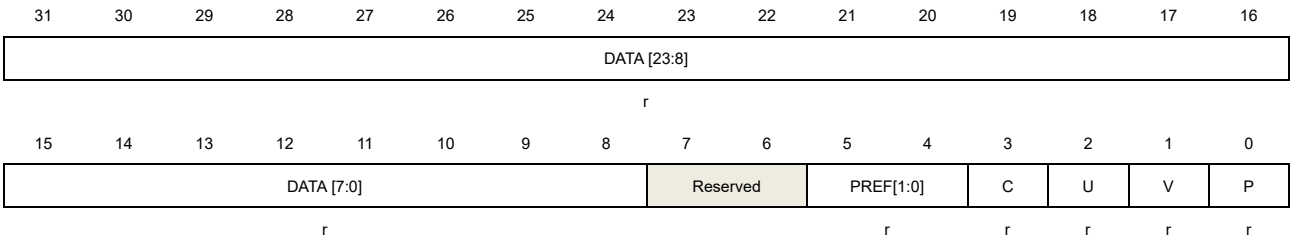
### RX data register format 0 (RSPDIF_DATA_F0)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:28 | PREF[1:0] | Preamble type bits<br>These bits indicate the preamble received.<br>00: not used<br>01: Preamble B received<br>10: Preamble M received<br>11: Preamble W received<br>Note that if PTNCPEN = 1, this field is forced to zero |
| 27 | C | Channel status bit<br>Contains the received channel status bit, if CUNCPEN = 0, otherwise it is forced to 0 |
| 26 | U | User bit<br>Contains the received user bit, if CUNCPEN = 0, otherwise it is forced to 0 |
| 25 | V | Validity bit<br>Contains the received validity bit if VNCPEN = 0, otherwise it is forced to 0 |
| 24 | P | Parity error bit<br>Contains a copy of PERR bit if PNCPEN = 0, otherwise it is forced to 0 |
| 23:0 | DATA[23:0] | Data value<br>Contains the 24 received data bits, right-aligned. |

When RXDF[1:0]=2'b01, the format is as follows:

### RX data register format 1 (RSPDIF_DATA_F1)




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:8 | DATA[23:0] | Data value<br>Contains the 24 received data bits, left-aligned. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | PREF[1:0] | Preamble type bits<br>These bits indicate the preamble received.<br>00: not used<br>01: Preamble B received<br>10: Preamble M received<br>11: Preamble W received<br>Note that if PTNCPEN = 1, this field is forced to zero |
| 3 | C | Channel status bit<br>Contains the received channel status bit, if CUNCPEN = 0, otherwise it is forced to 0 |
| 2 | U | User bit<br>Contains the received user bit, if CUNCPEN = 0, otherwise it is forced to 0 |
| 1 | V | Validity bit<br>Contains the received validity bit if VNCPEN = 0, otherwise it is forced to 0 |
| 0 | P | Parity error bit<br>Contains a copy of PERR bit if PNCPEN = 0, otherwise it is forced to 0 |

When RXDF[1:0]=2'b10, the format is as follows:

**RX data register format 2 (RSPDIF_DATA_F2)**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA2[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATA1[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

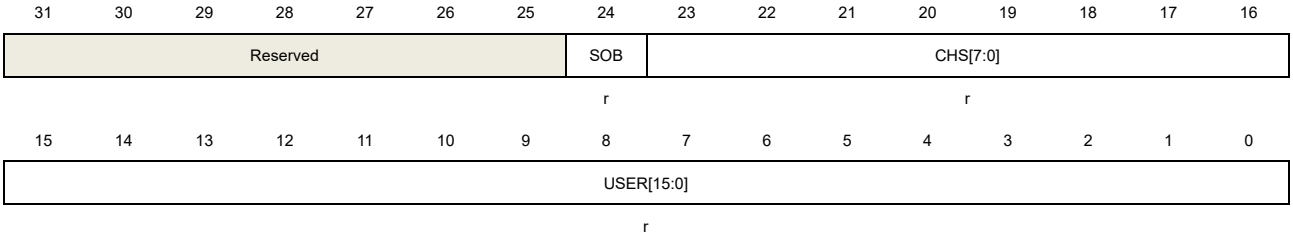
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | DATA2[15:0] | When in steo mode: this field contains the channel A data.<br>When in mono mode: this field contains the oldest value. |
| 15:0 | DATA1[15:0] | When in steo mode: this field contains the channel B data.<br>When in mono mode: this field contains the more recent value. |

### 33.4.6. RX Channel status register (RSPDIF_CHSTAT)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



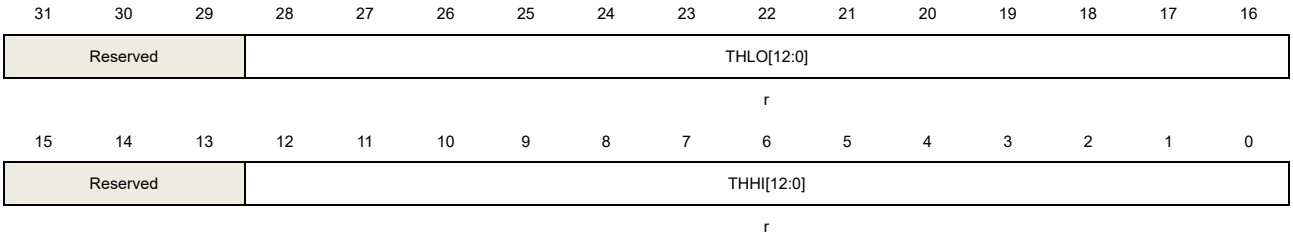

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | SOB | Start of block<br>This bit indicates if the bit CHS[0] corresponds to the first bit of a new block<br>0: CHS[0] is not the first bit of a new block<br>1: CHS[0] is the first bit of a new block |
| 23:16 | CHS[7:0] | Channel status information<br>Bit CHS[0] is the oldest value |
| 15:0 | USER[15:0] | User data information<br>Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B. So USR[n] bits come from channel A is n is even, otherwise they come from channel B. |

## 33.4.7. RX data threshold register (RSPDIF_DTH)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:16 | THLO[12:0] | Low threshold<br>THLO = 1.5 x UI / $T_{rspdif_ck}$<br>This field contains the current low threshold estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THLO is limited to a period of the rspdif_ck. The sampling rate can be estimated as follow:<br>Sampling Rate = [2 x THLO x $T_{rspdif_ck}$ +/- $T_{rspdif_ck}$ x 2/3 |

| | | |
|---|---|---|
| | | Note that THLO is updated by the hardware when SYNDO goes high, and then every frame. |
| 15:13 | Reserved | Must be kept at reset value. |
| 12:0 | THHI[12:0] | High threshold<br>THHI = 2.5 x UI / $T_{rspdif_ck}$<br>This field contains the current high threshold estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THHI is limited to a period of the rspdif_ck. The sampling rate can be estimated as follow:<br>Sampling Rate = [2 x THHI x $T_{rspdif_ck}$ +/- $T_{rspdif_ck}$] x 2/5<br>Note that THHI is updated by the hardware when SYNDO goes high, and then every frame. |

# 34. Serial Audio Interface (SAI) (For GD32H737_757_759)

## 34.1. Overview

The Serial Audio Interface (SAI) is designed to target a wide range of commonly used audio protocols, both in mono and stereo modes, such as I2S, PCM/DSP, AC'97, LSB or MSB-justified and TDM. The audio module can be output as SPDIF when configured in transmitter mode.

To realize these multitudes of configuration, two identically independent audio sub-blocks are implemented. Each audio sub-block contains up to 4 IO pins (SD, SCK, FS, and MCLK). Parts of these pins are designed to be shareable when the two audio sub-blocks are configured to be synchronous with each other.

The SAI can be configured to any of the master/slave and transmitter/receiver combination, full/half-duplex operating mode depends on synchronous/asynchronous configuration of the audio sub-blocks.

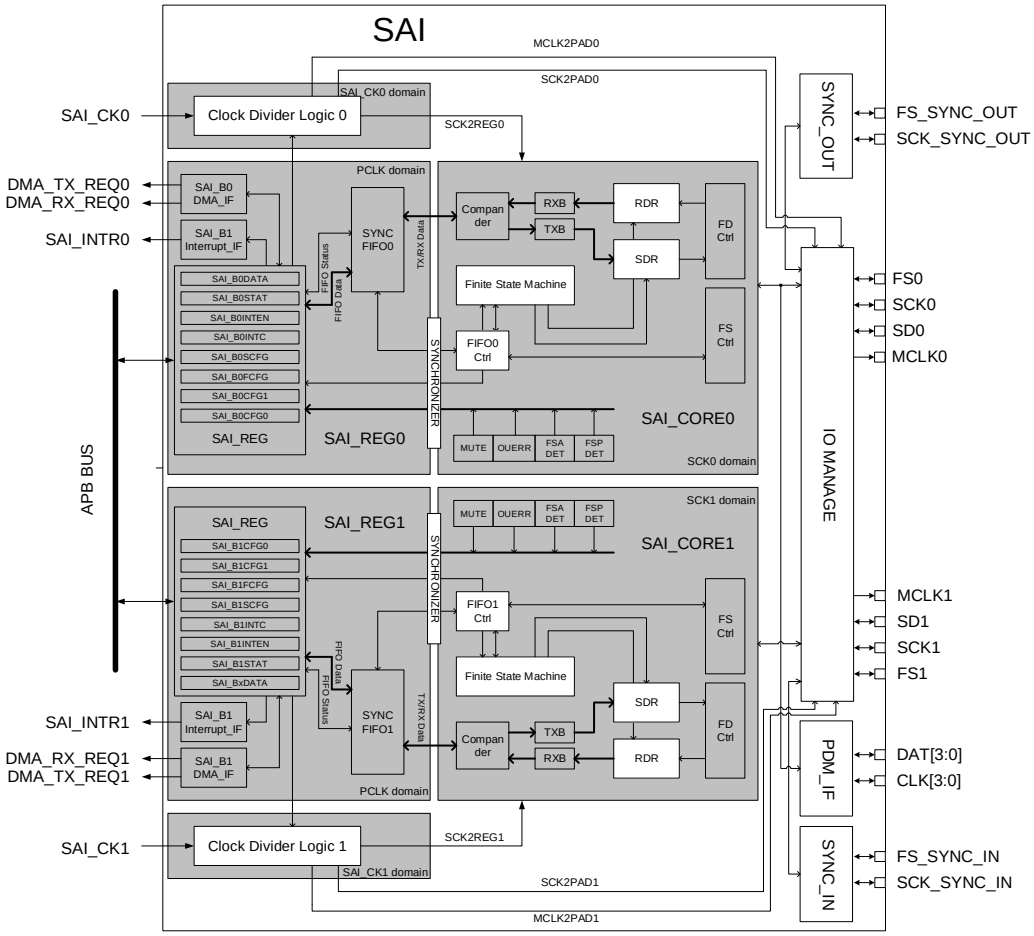
## 34.2. Characteristics

- Two independent audio sub-blocks.
- Each audio sub-block can be configured as any of the master/slave and transmitter/receiver combination with 8-word FIFO.
- Local clock divider logic to satisfy the various audio sampling rates.
- Flexible audio protocol configuration such as I2S, PCM/DSP, AC'97, LSB or MSB-justified and TDM.
- PDM interface, supporting up to 4 microphone pairs (GD32H73x_75x support 3 microphone pairs).
- Mono/Stereo audio capability with mute option.
- Frame Synchronization configuration (active level, active length and offset).
- Each audio frame contains up to 16 configurable slots.
- Slot length is flexible, and can be configured as active or inactive.
- Each slot can hold a data of size 8-, 10-, 16-, 20-, 24-, and 32-bits with configurable first bit offset, and configurable LSB or MSB data transfer.
- Serial clock strobe edge selection (SCK).
- Error flag and interrupt sources
  - FIFO overrun and underrun
  - Frame synchronization advanced detection in salve mode.
  - Frame synchronization postpone detection in slave mode.
  - AC'97 codec not ready.
  - Wrong clock configuration
- Two independent DMA interface for each audio sub-block. Support slave mode with a frequency up to 4MHz

## 34.3. Function overview

### 34.3.1. Block diagram

**Figure 34-1. block diagram**




**Note:** GD32H73x_75x PDM interface signals are only valid for DAT[2:0] and CLK[1:0].

The flexible audio transceiver is the integration of two identically independent sub-blocks, with an IO management module attached to their outputs. Each audio sub-block is composed of three isolated timing domain, the SAI_CLK, FCLK, and PCLK domain. Clock divider logic which defines the audio sampling rate resides in SAI_CLK domain, its clock output is wired to FCLK domain where SAI main functional state machine, compress/decompress, transition/reception logic and interrupt generation logic is. The main control registers and the synchronous FIFO is located in the PCLK domain. The synchronous FIFO can be accessed either by ARM CPU APB Bus or by DMA controller.

Each of the audio sub-block can be configured as any combination of master/slave and transmitter/receiver pair, the Frame Synchronization (FS) and Serial Clock (SCK) are generated in master mode, and received in slave mode from an external master or another audio sub-block in synchronous mode. The Master Clock (MCLK) generated only in master mode for external DAC/ADC operations. There is one exception that, when SAI is configured

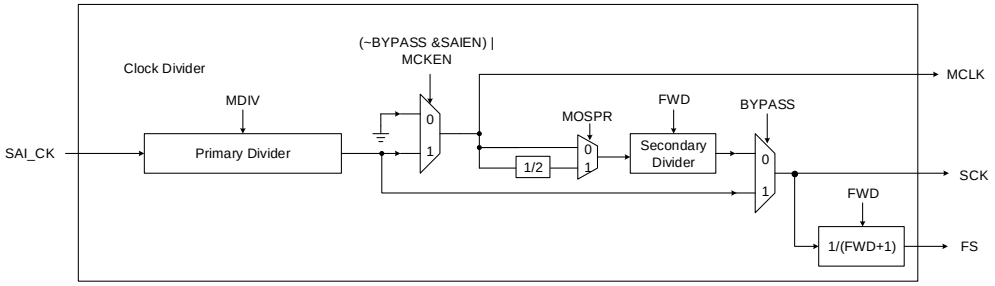
in AC'97 protocol, FS is forced to be an output, independent of master/slave configurations. The Serial Data (SD) IO pin is configured as output in transition and input in reception.

The IO Management block controls the IO pins of each audio sub-block, when the two sub-blocks are declared synchronous with each other, FS, SCK and MCLK can be shared, those pins of the synchronous sub-block are freed and can be used as general purpose IO.

### 34.3.2. Clock divider

The clock divider logic which is present in both SAI audio sub-blocks turns on only when they are configured as master devices, otherwise it is off and both MCLK and SCK output stays low. The divider's source clock SAI_CK is recommended to take on 2 specific values, 45.1584 MHz and 49.152 MHz to generate standard audio sampling rate. The clock divider logic is composed of a primary clock divider which is used to generate the required master clock (MCLK) and a secondary clock divider which is used to generate the bit clock (SCK), its block diagram is as follows.

**Figure 34-2. Clock divider logic**




The primary clock divider's ratio PDIV is a direct link to Master Clock Divider Ratio control field inside SAI control register, its output frequency could be obtained by the following formula.

$$f_{MCLK} = \begin{cases} \frac{f_{SAI_CK}}{MDIV}, & MDIV \neq 0 \\ f_{SAI_CK}, & MDIV = 0 \end{cases} \quad (34\text{-}1)$$

**Note:** The formula above holds only when BYPASS is not valid, SAI is on and MCKEN is on, otherwise, MCLK stays low level.

The secondary clock divider's ratio SDIV is connected to the Frame width (FWD) control field inside SAI frame configuration register, when BYPASS is off, the secondary divider's input is linked to the primary divider's output stage. The following formula determines the relationship between SAI_CK and bit clock (SCK) sampling rate.

$$f_{SCK} = \frac{f_{SAI_CK} \times (FWD + 1)}{MDIV \times (MOSPR + 1) \times 256} \quad (34\text{-}2)$$

The frame synchronization frequency is given by:

$$f_{FS} = \frac{f_{SAI_CK}}{MDIV \times (MOSPR + 1) \times 256} \quad (34\text{-}3)$$

When BYPASS is set, the master clock (MCLK) is turned off with a fixed output value of 0, while the bit clock (SCK) is linked directly to SAI_CK. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8.

When BYPASS is cleared, It is necessary to set (FWD + 1) equals to the result of an exponential function of base 2 in master mode to guaranty $T_{SCK}$ divisible by $T_{MCLK}$.

Some of the commonly used audio sampling rate configurations when frame width is 256-bit are listed in the ***Table 34-1. Commonly used audio sampling rate***.

**Table 34-1. Commonly used audio sampling rate**

| SAI_CK Clock Rate | Standard Audio Sampling Rate | Master Clock Divide Ratio |
|---|---|---|
| 192kHz x 256 | 192 kHz | MDIV = 1 |
| | 96 kHz | MDIV = 2 |
| | 48 kHz | MDIV = 4 |
| | 16 kHz | MDIV = 12 |
| | 8 kHz | MDIV = 14 |
| 44.1kHz x 256 | 44.1 kHz | MDIV = 1 |
| | 22.05 kHz | MDIV = 2 |
| | 11.025 kHz | MDIV = 4 |

### 34.3.3. Operating mode

The SAI audio sub-blocks can be configured in any combination of the master/slave and transmitter/receiver pair independently.

#### Master

Frame synchronization (FS) is always generated by the master at the start of a frame as long as the FIFO is not empty to indicate frame start or channel identification. Serial clock (SCK) and master clock (MCK) are other signals generated by the master, SCK is used exclusively by the slave, acting as its bit clock. Unlike FS, SCK and MCLK′s generation is not conditioned by the emptiness of the FIFO, they are generated as soon as the audio sub-block is enabled.

#### Slave

Slave receives FS and SCK signal from the master, its source depends on whether the audio sub-block is declared synchronous or asynchronous. When asynchronous mode is chosen, the FS and SCK signals source is connected directly to the chip level IO ports, while when synchronous mode is chosen, the FS and SCK signal is wired to another audio sub-block′s FS and SCK signal. Users must always enable the slave before the master; otherwise the slave will not receive the complete data from the master.

#### Transmitter

When the audio sub-block is configured as transmitter, serial data (SD) is an output. If the FIFO is still empty after the audio sub-block is enabled, a 0 value is sent, and the underrun

flag (OUERR) is raised.

#### Receiver

When the audio sub-block is configured as receiver, serial data (SD) is an input. Slave receiver always looks at the FS signal, when the first active edge is observed, it stores the received data, the coordination of the following data reception is handled by the internal finite state machine, and the receiver terminates reception at the end of a frame where a disable signal has already been set.

### 34.3.4. SAI synchronization mode

SAI synchronization mode includes two modes: internal synchronization mode and external synchronization mode.

#### Internal synchronization

The internal synchronization mode has the advantage of reducing the number of external pins occupied during communication. The SAI sub-modules SAI_B0 and SAI_B1 run synchronously, and they will share the SAI_FS and SAI_SCK signals, thereby releasing the GPIO pins of SCKx, FSx and MCLKx.

The SAI sub-module in internal synchronization mode can be configured in the following modes in full-duplex communication:

1. SAI_B0 (or SAI_B1) is configured as the master module, and SAI_B1 (or SAI_B0) is configured as the slave module.
2. Both SAI_B0 and SAI_B1 are configured as slave modules.
3. SAI_B0 (or SAI_B1) is configured as an asynchronous module, and SAI_B1 (or SAI_B0) is configured as a synchronous module.

**Note**: Due to the internal resynchronization phase, the frequency of the PCLK APB is required to be above twice the bit rate clock frequency.

#### External synchronization

External synchronization means that the SAI audio sub-module is synchronized with other SAIs. The SYNO[1:0] in the SAI_SYNCFG register is configured to determine the synchronization source that provides FS and SCK signals for other SAIs(SAI_B0 or SAI_B1). By configuring SYNI[1:0] in the SAI_SYNCFG register, it is determined which SAI is selected by the SAI receiving the synchronization signal for synchronization. By configuring the SYNCMOD[1:0] in the SAI_BxCFG0 register, specify whether the SAI audio sub-module is synchronized with other SAIs.

If both audio sub-modules in SAI need to be synchronized with another SAI, each audio sub-module can be configured to synchronize with another SAI module by configuring the SYNCMOD[1:0] bits or configure an audio sub-module to synchronize with another SAI module by configuring the SYNCMOD[1:0] bits.Then configure the other audio sub-modules

to synchronize with the second SAI audio sub-module by configuring the SYNCMOD[1:0] bits.

*Table 34-2. External synchronization configuration* is a reference for external synchronization configuration.

**Table 34-2. External synchronization configuration**

| SAI module | SYNI =2 | SYNI =1 | SYNI =0 |
|---|---|---|---|
| SAI0 | SAI2 sync. | SAI1 sync. | Reserved |
| SAI1 | SAI2 sync. | Reserved | SAI0 sync. |
| SAI2 | Reserved | SAI1 sync. | SAI0 sync. |

**Note**: When an SAI sub-module is configured in slave mode, there are synchronization signal direction limitations: the slave sub-module can only receive synchronization signals and cannot provide frame synchronization (FS) signals to other SAI sub-modules.

Special attention is required in the following scenarios:

**Internal synchronization mode**: The slave sub-module cannot serve as a synchronization signal source and can only receive FS and SCK signals from the master module.

**External synchronization mode**: If the slave module needs to synchronize with an external SAI, it must receive external FS and SCK signals through GPIO pins and cannot provide synchronization signals to external SAI through internal paths.

This limitation should be fully considered during the system design phase, with proper planning of the master-slave relationships and pin allocation for SAI modules..

### 34.3.5. Frame configuration

#### Frame synchronization

Frame synchronization is the coordination signal between master and slave to initiate a transfer. A number of parameters were implemented to manipulate with its waveform.

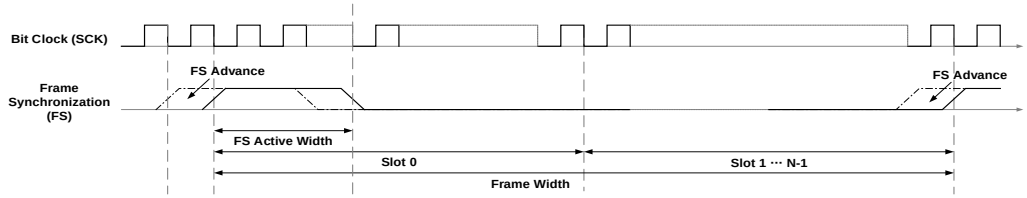
#### Frame synchronization advancement

Frame synchronization active edge could be aligned with the start of the first bit of the first slot, or one bit clock (SCK) cycle in advance, depending on the control field FSOST in SAI_BxFCFG register. *Figure 34-3 FS active width* shows how the FS waveform is changed.

#### Frame synchronization active width

Frame synchronization active width as shown in the following diagram depends on the control field FSAWD in SAI_BxFCFG register, its actual width equals (FSAWD + 1) SCK clock cycles. The minimum is 1 SCK clock cycle, while the maximum is 128 SCK clock cycles, which is half the maximum frame width. When FSFUNC is set, FS determines not only frame start, but also channel identification, (FSAWD+1) has to set equals half the frame width, otherwise the audio sub-block’s function is not guaranteed.

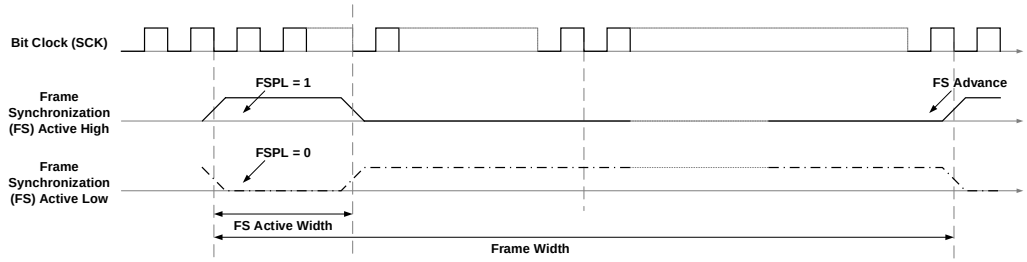
**Figure 34-3 FS active width**




## Frame synchronization polarity

Frame synchronization active level could be configured through FSPL control filed in SAI_BxFCFG register, as shown in the ***Figure 34-4 FS polarity***.

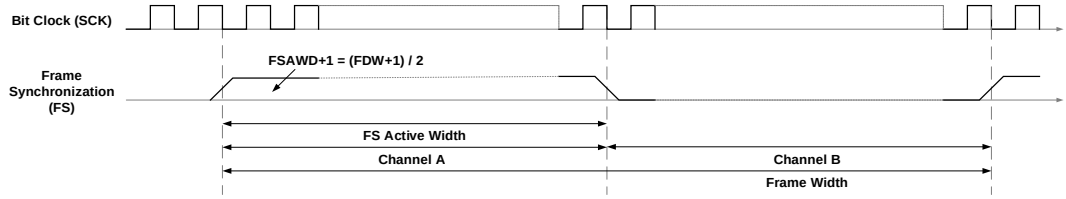
**Figure 34-4 FS polarity**




## Frame synchronization function

Frame synchronization function definition is configured though the FSFUNC in SAI_BxFCFG register. Two specific function could be selected, when FSFUNC is set to 1, FS not only represent frame start, but also channel number identification, in this case, frame active width (FSAWD + 1) should be configured to half of the frame width, as shown in ***Figure 34-5 FS function***, otherwise the audio sub-block behavior is not guaranteed. When FSFUNC is set to 0, FS only represents frame start.

**Figure 34-5 FS function**




## Frame width

Frame width cannot be lower than 8-bit, which corresponding to one byte of data, and cannot exceed 256 bits.

In master mode, if BYPASS is cleared, frame length (FWD+1) should be set to a value equals the result of a base 2 exponential function within 8 to 256 to guarantee an integer number of MCLK cycles within one SCK cycle, this is must for correct external DAC/ADC operations. Otherwise error clock flag (ERRCK) is raised in SAI_BxSTAT register, and an interrupt is generated if error clock interrupt enable (ERRCKIE) is set in SAI_BxINTEN register. While if

BYPASS is set, there will be no restriction on frame length configuration, master clock is automatically disabled.

In slave mode, frame length configuration is used to coordinate the internal finite state machine knowing the start and the end of an active frame. Another utility is used for advanced or postponed frame synchronization detection, an error flag is raised if this situation do occur and an interrupt is generated if the corresponding interrupt enable bit is set, refer to chapters ***Error flags*** and ***Interrupts*** for more details.

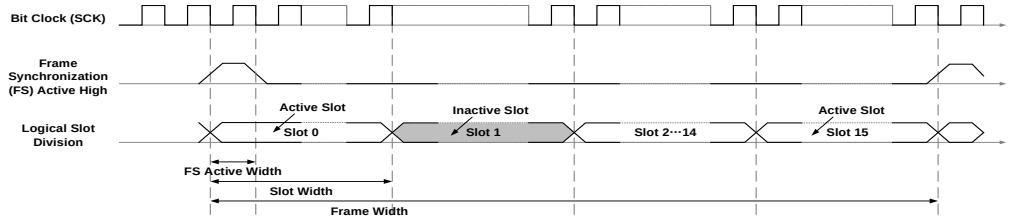
### 34.3.6. Slot configuration

Each SAI frame could be logically divided up to 16 slots, each slot's activation status and their distribution is further controlled through the configuration registers. Slot width can be configured to 16-, 32-bits or the same as data width through the SLOTWD control field in the SAI_BxSCFG register.

#### Slot activation

Each slot's activation status could be managed independently through slot activation vector (SLOTAV) in SAI_BxSCFG register. SLOTAV is a 16-bits wide control field, and each bit controls the corresponding slot's activation status. The logical division of slots could be shown in the ***Figure 34-6 Slot activation***.

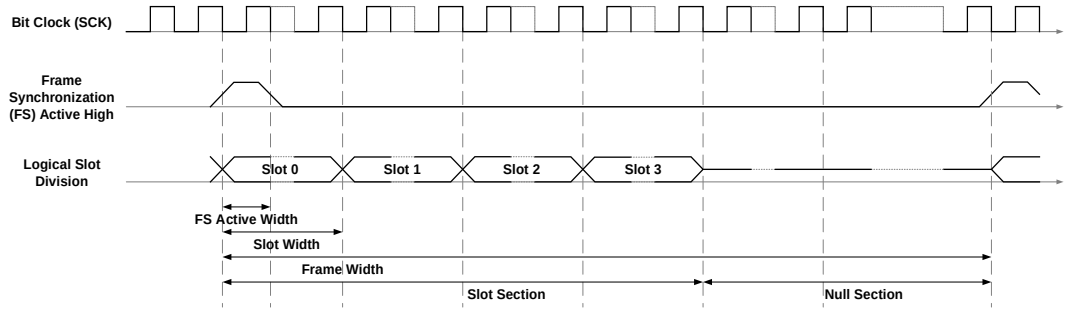
**Figure 34-6 Slot activation**




#### Slot distribution

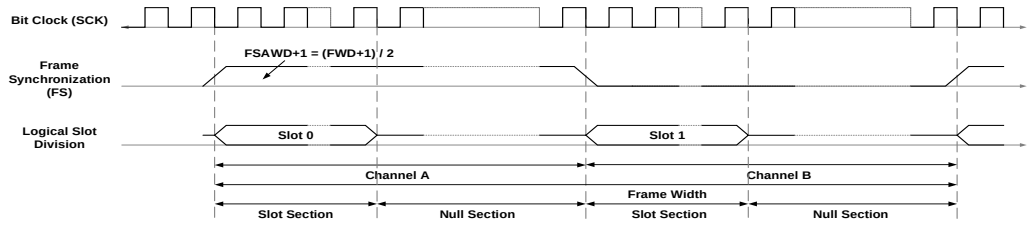
There can be not slot distribution under a specific circumstance that the product of slot number and slot width is less than frame width. Slot section is defined where there is slot distribution; otherwise, it is called a null section. When FSFUNC set to 0, where FS only signals frame start, the null section follows the last slot until the start of the next frame, as shown in the ***Figure 34-7 Slot distribution when FUNC = 0.***

**Figure 34-7 Slot distribution when FUNC = 0**




When FSFUNC = 1, where FS not only signals frame start, but also channel number identification, slot section and null section are evenly distributed to both channels. Null section fills the space between the last slots of the current channel till the start of the next channel's slot.

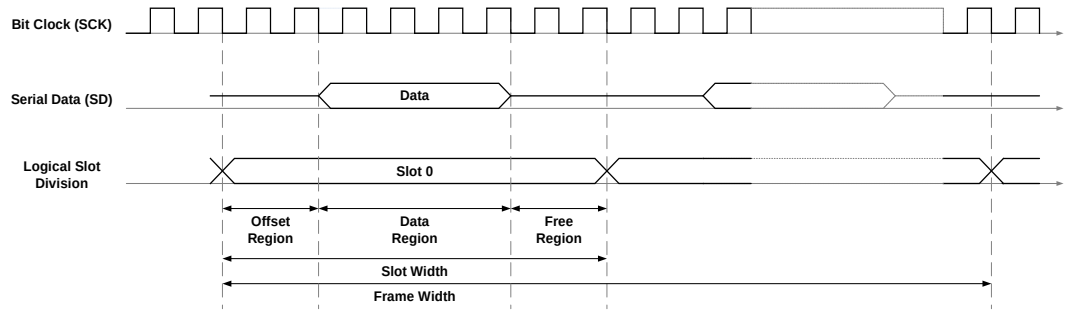
**Figure 34-8 Slot distribution when FUNC = 1**




## Serial data output management on inactive slots

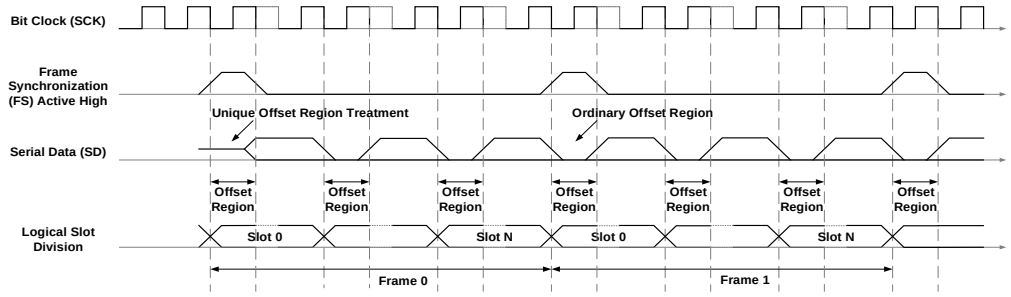
Serial data (SD) output behavior in the vicinity of inactive slots could be set according to the management policy defined in the serial data output mode (SDOM) bit in SAI_BxSCFG register, either SAI releases or drives a 0 value to the output. There are three special cases of SD output behavior, offset and free region needs special attention. Slot partition convention, namely offset, data and free region used in this manual is depicted in the ***Figure 34-9 Slot partition convention***.

**Figure 34-9 Slot partition convention**




Firstly, SD output during the offset region is decided by SDOM, if SDOM is set to 1, SAI will release the output, else a 0 is sent on the SD output, the diagram ***Figure 34-10 Offset region treatment*** shows the differences.

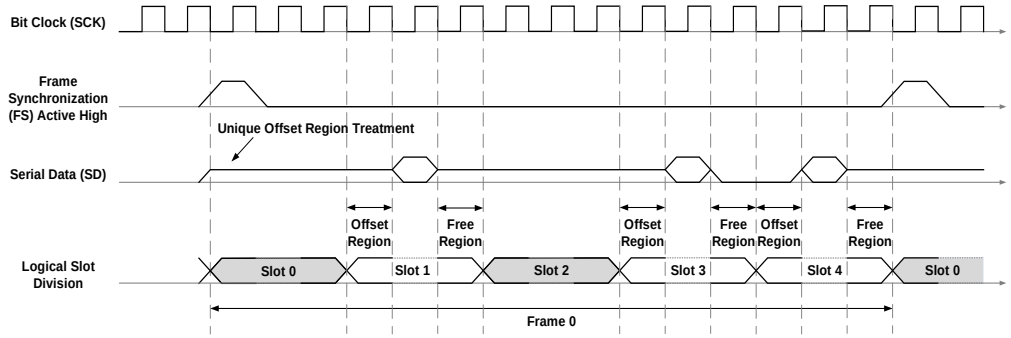
**Figure 34-10 Offset region treatment**




Secondly, SD output during the free region of the last slot has its reference on the activation status of the first slot, when SDOM is set, if slot 0 is inactive, SD output will be released, else a 0 is sent on the SD output. When SDOM is cleared, SD output is low independent of other slots activation status.

Lastly, SD output behavior during the offset and free region of slots in the middle of a frame has its reference on their next slot's activation status. If the following slot is inactive, and free region is present, SD output will be release when SDOM set to 1, and drive to 0 if not. SD output behavior of offset and free region near active and inactive slot is shown in the ***Figure 34-11 SD output management***.

**Figure 34-11 SD output management**




## 34.3.7. Data configuration

Data width is also flexible, it can take on the value of 8-, 10-, 16-, 20-, 24-bits and 32-bits wide, which is configured through the data width (DATAWD) control field in the SAI_BxSCFG register. Data inside an active slot could be shifted backward by setting the data offset (DATAOST), also located in the SAI_BxSCFG register. As described in the SD output management section (refer to ***Note:*** When an SAI sub-module is configured in slave mode, there are synchronization signal direction limitations: the slave sub-module can only receive synchronization signals and cannot provide frame synchronization (FS) signals to other SAI sub-modules.

Special attention is required in the following scenarios:

**Internal synchronization mode**: The slave sub-module cannot serve as a synchronization signal source and can only receive FS and SCK signals from the master module.

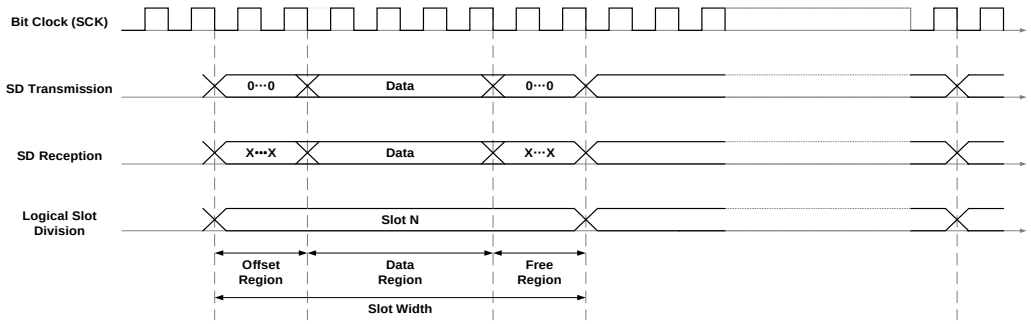
**External synchronization mode**: If the slave module needs to synchronize with an external

SAI, it must receive external FS and SCK signals through GPIO pins and cannot provide synchronization signals to external SAI through internal paths.

This limitation should be fully considered during the system design phase, with proper planning of the master-slave relationships and pin allocation for SAI modules..

Frame configuration), the space between the start of a slot and the first bit of the data is called the offset region, and the space between the last bit of the data and the end of the slot is the free region. When the audio sub-block is configured as transmitter, and offset or free region is present, a 0 is sent through the SD output, the actual behavior of the SD line not only depends on the output value, but also on the line management condition and the slot activation status of nearby slots. When the audio sub-block is configure as receiver, and offset or free region is present, data reception during these regions will be ignored. Data transmission and reception are shown in ***Figure 34-12 Data configuration***.

**Figure 34-12 Data configuration**




## 34.3.8. Internal FIFO

An 8-word depth internal FIFO is implemented inside each SAI audio sub-block independently to increase transaction efficiency, these FIFO could be accessed by both CPU and DMA. FIFO request interrupt mechanism used to call for CPU and DMA access, request is raised depending on operating mode, FIFO threshold, FIFO status, and DMA burst size. FIFO request interrupt generation is summarized in the ***Table 34-3 FIFO request generation conditions*** and cleared if the underlying conditions are not satisfied.

**Table 34-3 FIFO request generation conditions**

| Transmitter: OPTMOD[0] = 0 | | | | Receiver: OPTMOD[0] = 1 | | | |
|---|---|---|---|---|---|---|---|
| FIFO Threshold | FFTH | FIFO Status | FFSTAT | FIFO Threshold | FFTH | FIFO Status | FFSTAT |
| Empty | = 000 | Empty | = 000 | Empty | = 000 | Not Empty | ≥ 001 |
| 1/4 Full | = 001 | <1/4 Full | <010 | 1/4 Full | = 001 | ≥ 1/4 Full | ≥ 010 |
| 1/2 Full | = 010 | <1/2 Full | <011 | 1/2 Full | = 010 | ≥ 1/2 Full | ≥ 011 |
| 3/4 Full | = 011 | <3/4 Full | <100 | 3/4 Full | = 011 | ≥ 3/4 Full | ≥ 100 |
| Full | = 100 | Not Full | <101 | Full | = 100 | Full | = 101 |

FIFO flush is done through the FLUSH control field in SAI_BxCFG1 register, when FLUSH is set, all the data content in the FIFO will be cleared, and read/write pointer reset to 0.

**Note:** DMA request generation is implicitly depended on FIFO request, detailed information will be provided in DMA interface section.

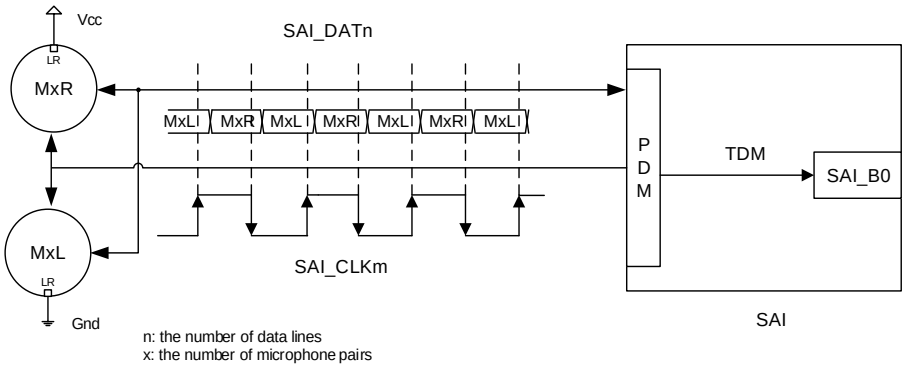
### 34.3.9. PDM interface

Digital microphones can realize data transmission through the PDM interface. The PDM interface supports up to 4 pairs of digital microphones in parallel.

#### Timing and connection

The schematic diagram of microphones connected by PDM interface is shown in the ***Figure 34-13. PDM typical connection and timing***.

**Figure 34-13. PDM typical connection and timing**




**Note:** GD32H73x_75x PDM interface signals are only valid for DAT[2:0] and CLK[1:0].

The microphone LR pin connected to VCC is the left microphone channel, and the one connected to GND is the right microphone channel. The left channel and the right channel microphone share one clock line SAI_CLK. The left channel data is sampled on the rising edge, and the right channel data is sampled on the falling edge to complete the two-channel data collection.

**Note:** The PDM interface can only be used in connection with SAI_B0.

#### PDM interface enable

The PDM interface enabling process is as follows:

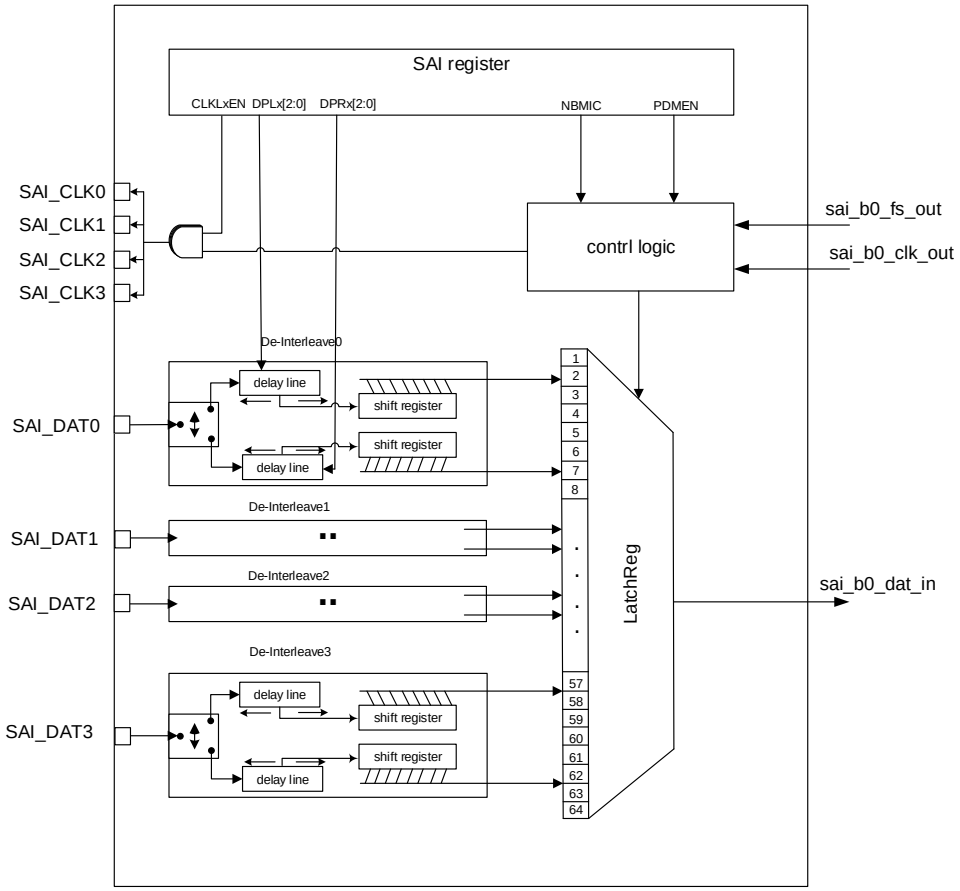
1. Configure TDM as master mode, OPTMOD[0:1] of SAI_B0CFG0 register is host receiving mode, and PROT[1:0] bits are free protocol.
2. Configure the PDM interface: select the number of microphones by configuring the MICNUMSEL[1:0] bits of the SAI_PDMCTL register, and configure the CLKLx bit in the SAI_B0CFG0 register to enable the clock.
3. Configure the PDMEN bit in the SAI_PDMCTL register to enable the PDM interface.
4. Configure the SAIEN bit in the SAI_B0CFG0 register and enable SAI_B0.

### Data processing

The data processing sequence in PDM is as follows:

1. SAI_B0 generates the clock to generate the bit stream clock via the TDM link to the PDM interface
2. By configuring the DPLx[2:0] and DPRx[2:0] bits of the SAI_PDMCFG register, the PDM interface is implemented to interleave and delay the bit stream data SAI_DATx generated by the microphone, thereby adjusting the delay generated by the microphone.
3. The shift register converts the data stream into bytes, and transmits the data to SAI_B0 through the TDM link.

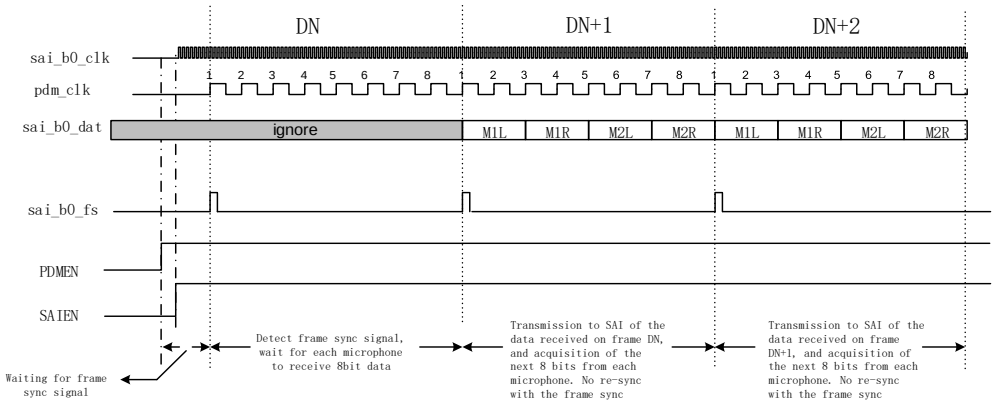
**Figure 34-14. PDM data processing diagram**




### Data transfer start process

After the PDM interface is enabled, the microphone data sampling starts after the frame synchronization event occurs. After 8 SAI_CK clocks, the microphone data will be transmitted to the SAI through the TDM interface. The ***Figure 34-15. The start-up process of PDM data transmission*** shows the start-up process of PDM data transmission..

**Figure 34-15. The start-up process of PDM data transmission**




## Data format

The data format from the microphone data is mainly related to the following configuration,

1. Configure MICNUMSEL[1:0] in the PDMCTL register to set the number of microphones.
2. Configure SLOTWD[1:0] in the BxSCFG register to set the slot width.
3. Configure the SHIFTDIR bit to set the MSB/LSB of the transmitted data.

**Table 34-4 Get the microphone data read register times under different configurations**

| Number of microphones | Slot width | The number of times to read the BxDATA register to obtain the microphone data |
|---|---|---|
| 8 | 32 | 2 |
| | 16 | 4 |
| | 8 | 8 |
| 4 | 32 | 1 |
| | 16 | 2 |
| | 8 | 4 |
| 2 | 16 | 1 |
| | 8 | 2 |

**Note:** GD32H73x_75x supports 3 pairs of microphones.

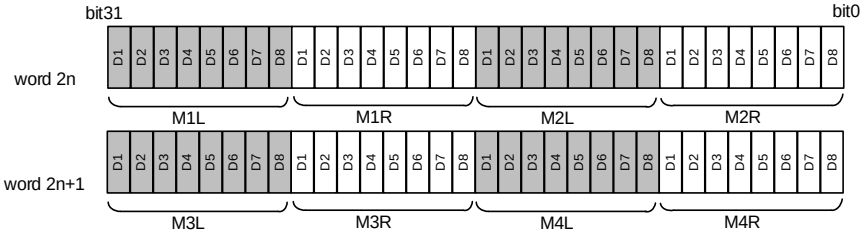
The following figure shows the schematic diagram of register data arrangement under different number of microphones and slot width.

**Figure 34-16. Eight microphones, under different slot widths, BxDATA register data**

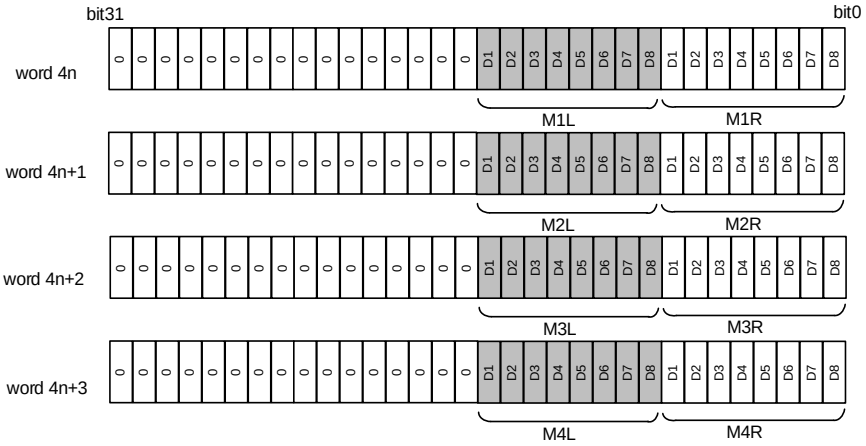
**format**

Eight microphone configuration

32-bit slot width SHIFTDIR = 0




16-bit slot width SHIFTDIR = 0




8-bit slot width SHIFTDIR = 0

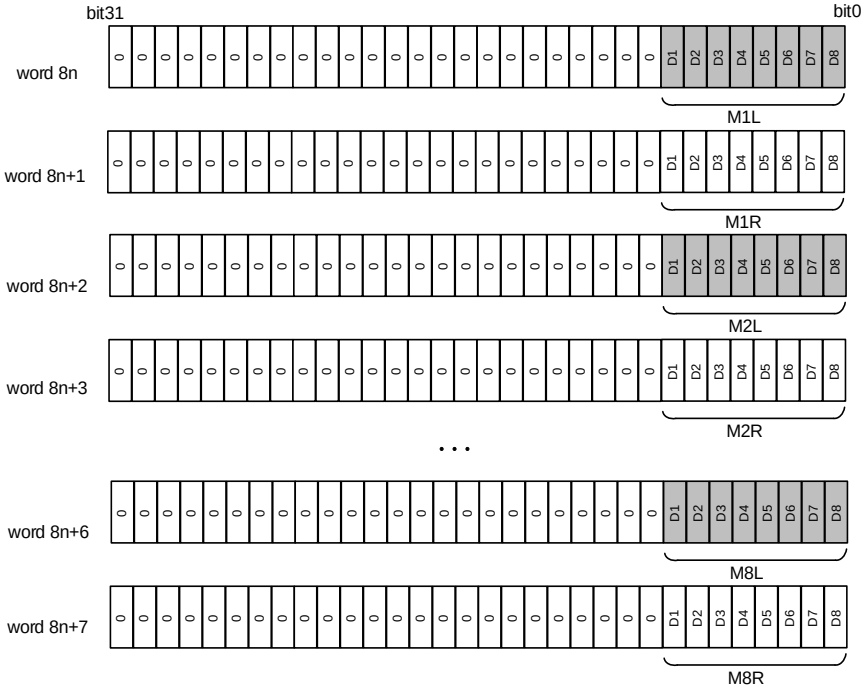
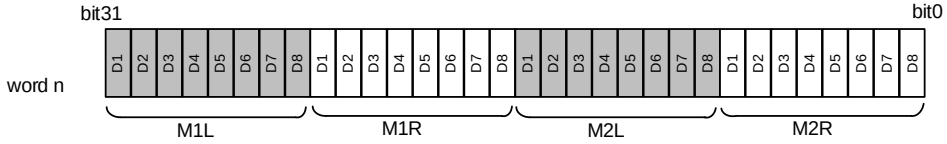



**Figure 34-17. Four microphones, under different slot widths, BxDATA register data**

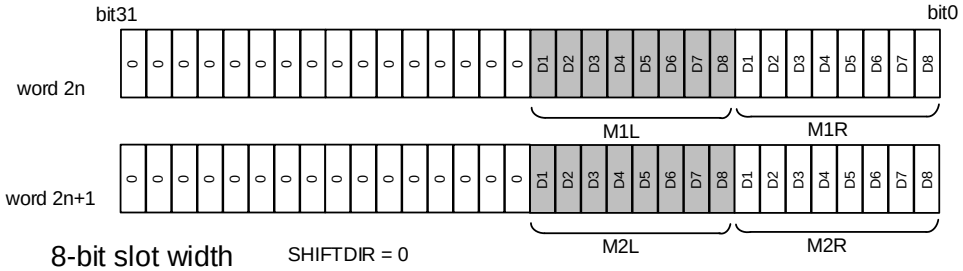
**format**

Four microphone configuration

32-bit slot width SHIFTDIR = 0




16-bit slot width SHIFTDIR = 0




8-bit slot width SHIFTDIR = 0

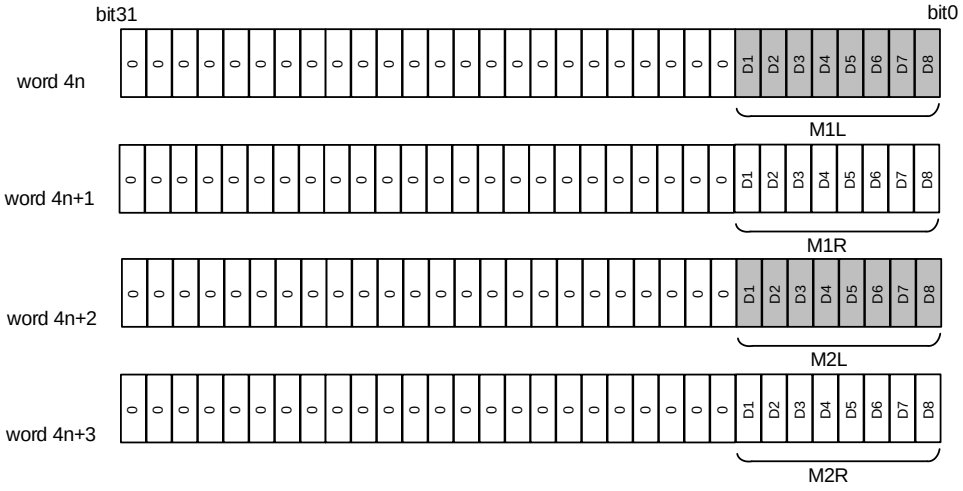
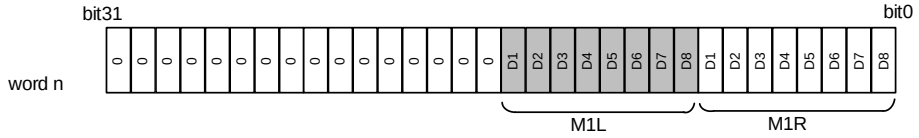



**Figure 34-18. Dual microphones, under different slot widths, BxDATA register data**

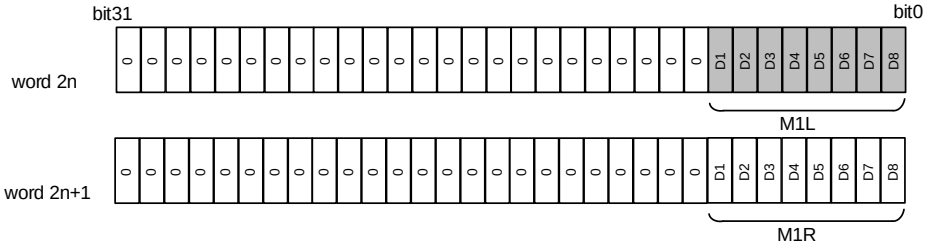
**format**

Dual microphone configuration

16-bit slot width SHIFTDIR = 0




8-bit slot width SHIFTDIR = 0




## TDM configuration

For the TDM configuration items of the PDM interface, please refer to ***Table 34-5. TDM configuration***.

**Table 34-5. TDM configuration**

| Registers | Bit fields | Values | Descriptions |
|---|---|---|---|
| SAI_B0CFG0 | OPTMOD[1:0] | 0b01 | Mode must be MASTER receiver |
| | PROT[1:0] | 0b00 | Polymorphic for TDM |
| | DATAWD[2:0] | x | Data width. |
| | SHIFTDIR | x | Data is shifted with MSB or LSB first. |
| | SAMPEDGE | 0 | Data is sampled on SCK falling or rising edge. |
| | MONO | 0 | Stereo or Mono mode selection. |
| | BYPASS | 1 | Clock divider logic bypass. |
| | MDIV[5:0] | x | Master clock divider ratio. |
| SAI_B0FCFG | FWD[7:0] | x | Frame width |
| | FSAWD[6:0] | 0 | Frame synchronization active width. |
| | FSFUNC | 0 | FS only defines frame start. |
| | FSPL | 1 | FS high active polarity. |
| | FSOST | 0 | FS active edge asserted at the beginning of the first bit of the first slot. |
| SAI_B0SCFG | DATAOST[4:0] | 0 | Data offset |
| | SLOTWD[1:0] | 0 | Slot width equals' data width |
| | SLOTNUM[3:0] | x | Slot number within frame |
| | SLOTAV[15:0] | x | To be adjusted according to SLOTNUM |

**Note:** When configuring PDM, the clock frequency, frame length, and slot size need to comply with the following three requirements:

1. The clock frequency configuration follows the following formula：

$$f_{SCK_B0} = f_{PDM_CLK} * (MICNUMSEL+1) * 2 \quad (34\text{-}4)$$

2. The frame width configuration follows the following formula:

$$FWD = (16 * (MICNUMSEL+1)) - 1 \quad (34\text{-}5)$$

3. Slot width needs to be configured as an integer multiple of (FWD+1).

### 34.3.10. AC’97 link controller

AC’97 link controller mode is selected via the PROT configuration in the SAI_BxCFG0 register. When this protocol is selected, many of the configuration field are ignored, include data shift direction, data width, most of frame and slot configurations, and part of interrupt control fields. The detail could be seen in the registers definition section.

AC’97 has a constant frame length of 256-bits, divided into 13-slots, the first slot is fixed at 16-bits width, and the rest 12 is fixed at 20-bits wide. User’s must set the data width (DATAWD)control filed in the SAI_BxCFG0 register to 16- or 20-bits wide, otherwise the audio sub-block’s behavior is not guaranteed.

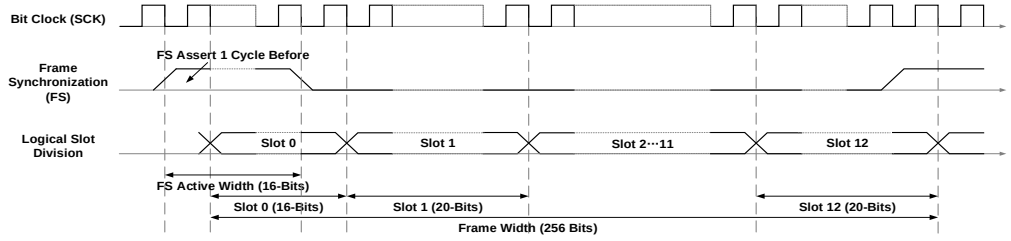
Bit2 of the TAG (slot 0) is reserved, no matter what value is written into the TAG, bit2 of the send data is 0.

Bit3 to 14 of the TAG (slot 0) act as the slot activation vector (SLOTAV) in free protocol, where the TAG slot (slot 0) is always active, bit3 correspond to slot 12, and bit14 corresponds to slot 1.

Bit15 of the TAG (slot 0) is codec ready status indication, when audio sub-block is configure as receiver, and a TAG received with Bit 15 low, indicate an audio codec not ready (ACNRDY) status, and ACNRDY flag is set accordingly. An interrupt is generated if both ACNRDY flag and audio codec not ready interrupt enable (ACNRDYIE) are both set.

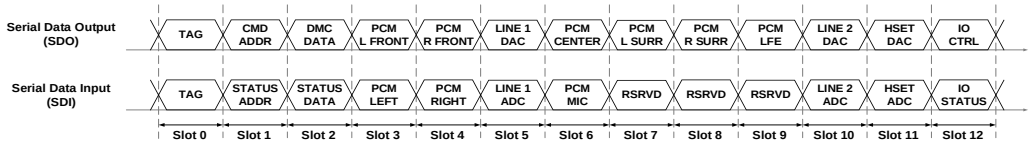
Frame synchronization active edge is asserted 1 clock cycle before the first bit of data, as shown in the ***Figure 34-19 AC'97 slot partition***.

**Figure 34-19 AC'97 slot partition**




The ***Figure 34-20 AC'97 tag definition*** give an overview of AC’97 slot partition.

**Figure 34-20 AC'97 tag definition**




The ***Table 34-6 AC'97 transmitter tag definition*** summarizes each slot's definition and meaning.

When AC'97 link controller acting as transmitter.

**Table 34-6 AC'97 transmitter tag definition**

| Slot | Name | Description |
|---|---|---|
| 0 | SDO TAG | MSBs indicate which slot contain valid data; LSBs convey Codec ID |
| 1 | Control CMD ADDR write port | Read/write command bit plus 7-bit Codec register address |
| 2 | Control DATA write port | 16-bit command register write data |
| 3,4 | PCM L&R DAC playback | 16-, 18-, 20-bit PCM data for Left and Right channels |
| 5 | Modem Line 1 DAC | 16-bit modem data for modem line 1 output |
| 6,7,8,9 | PCM Center, Surround L&R, LFE | 16-, 18-, 20-bit PCM data for Center, Surround L&R, LFE channels |
| 10 | Modem Line 2 DAC | 16-bit modem data for modem Line 2 output |
| 11 | Modem handset DAC | 16-bit modem data for modem Handset output |
| 12 | Modem IO control | GPIO write port for modem control |
| 10-11 | SPDIF Out | Optional AC-link bandwidth for SPDIF output |
| 6-12 | Double rate audio | Optional AC-link bandwidth for 88.2 or 96kHz on L, C, R channels. Actual slots used are controlled by the DRSS bits. |

When AC'97 link controller acting as receiver.

**Table 34-7 AC'97 receiver tag definition**

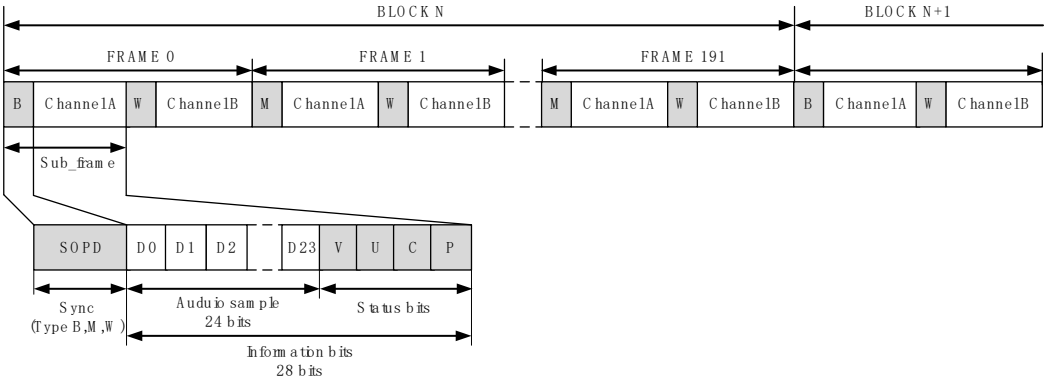
| Slot | Name | Description |
|---|---|---|
| 0 | SDI TAG | MSBs indicate witch slots contain valid data |
| 1 | STATUS ADDR read port | MSBs echo register address; LSBs indicate which slots request data |
| 2 | STATUS DATA read port | 16-bit command register read data |
| 3,4 | PCM L&R ADC record | 16-, 18- or 20-bit PCM data from Left and Right inputs |
| 5 | Modem Line 1 ADC | 16-bit modem data from modem Line 1 input |
| 6 | Dedicated Microphone ADC | 16-, 18- or 20-bit PCM data from output 3rdADC input |
| 7,8,9 | Vendor reserved | Vendor specific (enhanced input for docking, array mic, etc.) |
| 10 | Modem Line 2 ADC | 16-bit modem data from modem Line 2 input |
| 11 | Modem handset input ADC | 16-bit modem data form modem Handset input |
| 12 | Modem IO status | GPIO read port for modem status |

### 34.3.11. SPDIF Output

SPDIF (Sony/Philips Digital Interface) is a type of digital audio interconnect used in consumer audio equipment to output audio over reasonably short distances. SPDIF is standardized in IEC 60958.

*Figure 34-21 SPDIF data formate* shows SPDIF block formate and sub_frame formate.

**Figure 34-21 SPDIF data formate**




Each SPDIF block contains 192 frames of data, and each frame is composed of a left channel subframe (32 bits) and a right channel subframe (32 bits). Each subframe is composed of 4bit SOPD mode, 24bit data information and 4bit status information.

SOPD mode coding reference *Table 34-8. SOPD mode*

**Table 34-8. SOPD mode**

| Preceding state | 0 | 1 | Description |
|---|---|---|---|
| Preamble | Channel coding | | |
| B | 11101000 | 00010111 | Channel A data at the start of block |
| W | 11100100 | 00011011 | Channel B data at somewhere in the block |
| M | 11100010 | 00011101 | Channel A data |

The data filling of SPDIF data transmission in the SAI_BxDATA register should follow: SAI_BxDATA[26:24] contains the channel status bit, user bit and validity bit, SAI_BxDATA[23:0] contains the 24-bit data of the channel under consideration.

**Note:** If the data size is 20/16 bits, the data should be mapped to SAI_BxDATA[23:4] / SAI_BxDATA [23:8].

By configuring the OPTMOD[1] bit in the SAI_BxCFG0 register to 0, the master mode is forced to be selected. At the same time, the DATAWD[2:0] data bit width setting in the SAI_BxCFG0 register will be ignored and forced to be set to 24 bits. The symbol rate is configured through the clock generator. And encoded through the Manchester protocol.

The SAI first sends the adequate preamble for each sub-frame in a block. The SAI_BxDATA is then sent on the SD line (manchester coded). The SAI ends the sub-frame by transferring the Parity bit calculated as described in *Table 34-9 Parity bit calculation*.

**Table 34-9 Parity bit calculation**

| SAI_BxDATA [26:0] | Parity bit P value transferred |
|---|---|
| Odd number of 0 | 0 |
| Odd number of 1 | 1 |

For the SPDIF generator, the SAI shall provide a bit clock equal to twice of the symbol-rate.

More generally, the relationship between the audio sampling rate (FS) and the bit-clock rate (FSCK_x) is given by the formula：

$$F_s = \frac{F_{SCK_x}}{128} \quad (34\text{-}6)$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{SAI_CK_x}}{MDIV} \quad (34\text{-}7)$$

**Note:** The above formulas are valid only if BYPASS is set to 1 in SAI_BxCFG0.

### 34.3.12. Stereo/Mono

The SAI audio sub-block could be switched between stereo mode and mono mode by the MONO configuration in SAI_BxCFG0 register, note that when mono mode is selected, slot number must be configured to 2, else the audio sub-block behavior is not guaranteed.

When the audio sub-block is configured as transmitter, data sent during the first slot (slot 0) will be duplicated to the second slot (slot 1), in this case, the FIFO is access half as much as in stereo mode.

When the audio sub-block is configured as receiver, data received during the first slot is pushed into the FIFO, and data received during the second slot is discarded.

### 34.3.13. Mute

Users could set the mute property anywhere during an on-going frame through the MT bit in the SAI_BxCFG1 register, but mute will only take effect on the state of the next frame.

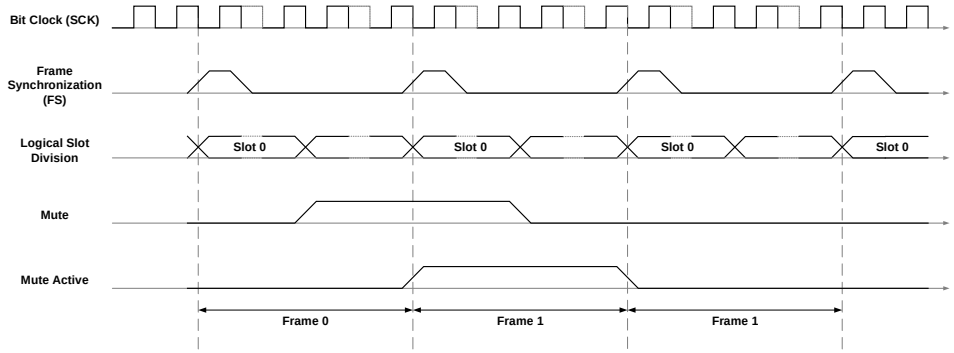
When the SAI audio sub-block is configured as transmitter and mute is configured, data is still read from the FIFO, put into the shift register as usual when mute take effect at next frame, the only difference is that the SD output if forced to a specific value determined by the mute value (MTVAL) configuration also located in the SAI_BxCFG1 register. When MTVALis set to 0, a 0 value is forced on the SD output during a mute frame. On the contrary, when MTVAL is set to 1, SD output behavior is further depended upon the slot number (SLOTNUM) configuration. When slot number is less or equals to 2, frame contents transmitted before mute active is repeated in mute frames. When slot number is greater than 2, SD output is forced to 0.

SAI audio sub-block that is configured in receiver mode could detect mute frame and generate

interrupt accordingly. A mute frame counter is implemented in each audio sub block, a frame received with each active slot's data equals to 0 will be regarded as a mute frame, and the internal mute frame counter is incremented. This mute frame counter is reset when SAI audio sub-block is disabled or when a frame is not a mute frame. If the number of successive mute frame received amount to a number equals to the mute frame count (MTFCNT) settings in the SAI_BxCFG1 register, the mute detection flag (MTDET) will be set in the SAI_BxSTAT register, and an interrupt is generated if mute detection interrupt enable (MTDETIE) is turned on in the SAI_BxINTEN register.

Mute frame activation is shown in the ***Figure 34-22 Mute frame activation***.

**Figure 34-22 Mute frame activation**




SD output behavior under different configuration is summarized in the ***Table 34-10 Mute frame composition***.

**Table 34-10 Mute frame composition**

| Slot Number | Mute Value = 1 | Mute Value = 0 |
|---|---|---|
| ≤ 2 | Frame before Mute active is repeated on SD Output | Forced to 0 |
| >2 | Forced to 0 | Forced to 0 |

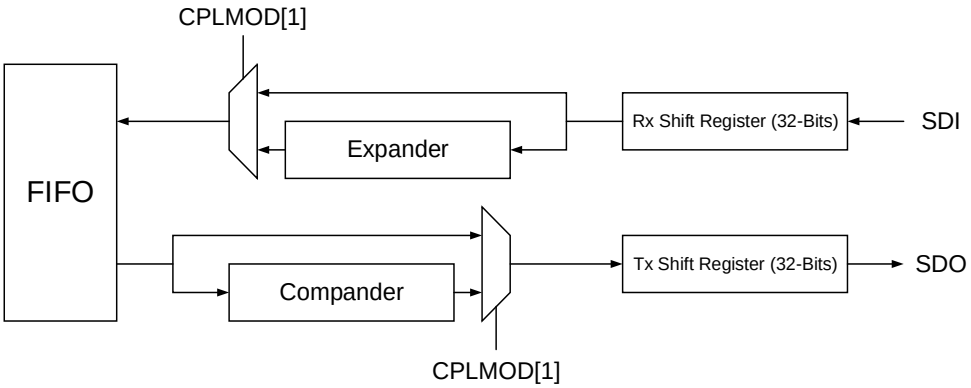
## 34.3.14. Compander

Compander is simply a system in which information is first compressed, transmitted through a bandwidth limited channel, and expanded in the receiving end. It is frequently used to reduce the bandwidth requirement for transmitting telephone quality speech, by reducing the 13-bitto 8-bit code words. Two international standards for encoding signal data to 8-bit codes are A-law and Mu-law. A-law is the accepted European standard, while Mu-law is the accepted standard in the United States and Japan.

Both A-law and Mu-law in implemented by the SAI, its configuration is located in the SAI_BxCFG1 register. The audio sub-block chooses between compressing and expanding according to the operating mode (OPTMOD). When the audio sub-block is configured as transmitter, compressing is selected. On the contrary, when configured as receiver, expanding mode is choose. Users can select between 1's or 2's complement as their default data representation as long as he has informed the SAI of his choice through the complement mode (CPLMOD) configuration in SAI_BxCFG1 register. In transmitter mode, no matter what

compander mode is selected, the hardware first convert the complement representation to signed magnitude representation, and the feed into the compander. While in receiver mode, the linear output data is converted form sign-magnitude to complement representation, and then stored in the FIFO.

**Figure 34-23 Compander data path**




## A-law compander

A-law is the CCITT recommended companding standard used across Europe, Limiting the linear sample values to 12 magnitude bits. The following chart illustrates the A-law encoding algorithm, the linear input data is in the signed-magnitude representation, with S representing the sign, and the following 12-bit representing the magnitude. The encoded output is 8-bit wide, with MSB representing the sign, not the sign bit S on both side of the ***Table 34-11 A-law encoding*** take on the same value. A, B, C, and D takes on the value of 0 or 1, and X represent don't cares.

**Table 34-11 A-law encoding**

| Linear Input Data | | | | | | | | | | | | | A-law Encoded Output | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | B | C | D | X | S | 0 | 0 | 0 | A | B | C | D |
| S | 0 | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | S | 0 | 0 | 1 | A | B | C | D |
| S | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | X | S | 0 | 1 | 0 | A | B | C | D |
| S | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | X | X | S | 0 | 1 | 1 | A | B | C | D |
| S | 0 | 0 | 0 | 1 | A | B | C | D | X | X | X | X | S | 1 | 0 | 0 | A | B | C | D |
| S | 0 | 0 | 1 | A | B | C | D | X | X | X | X | X | S | 1 | 0 | 1 | A | B | C | D |
| S | 0 | 1 | A | B | C | D | X | X | X | X | X | X | S | 1 | 1 | 0 | A | B | C | D |
| S | 1 | A | B | C | D | X | X | X | X | X | X | X | S | 1 | 1 | 1 | A | B | C | D |

After the input data is encoded through the logic defined in the table, an inversion pattern is applied to the 8-bit code to increase the density of transitions on the transmission line, a benefit to hardware performance. The inversion pattern is applied by XOR'ing the 8-bit code with 0x55.

Decoding the A-law encoded data is essentially a matter of reversing the steps in the encoding. The ***Table 34-12 A-law decoding*** illustrate the A-law decoding algorithm, applied after reversing the inversion pattern. The least significant bits discarded in the encoding process

are approximated by the median value of the interval. This is shown in the output section by the trailing 1…0 pattern after the D bit.

**Table 34-12 A-law decoding**

| A-law Encoded Input | | | | | | | | Linear Output data | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | 0 | 0 | 0 | A | B | C | D | S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | A | B | C | D | 1 |
| S | 0 | 0 | 1 | A | B | C | D | S | 0 | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 |
| S | 0 | 1 | 0 | A | B | C | D | S | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 |
| S | 0 | 1 | 1 | A | B | C | D | S | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 |
| S | 1 | 0 | 0 | A | B | C | D | S | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 | 0 |
| S | 1 | 0 | 1 | A | B | C | D | S | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 | 0 | 0 |
| S | 1 | 1 | 0 | A | B | C | D | S | 0 | 1 | A | B | C | D | 0 | 0 | 0 | 0 | 0 | 0 |
| S | 1 | 1 | 1 | A | B | C | D | S | 1 | A | B | C | D | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

## Mu-Law compander

The Unite States and Japan use the Mu-law companding, limiting the linear sample value to 13 magnitude bits. The encoding and decoding process for Mu-law is similar to A-law. There are, however, a few notable differences:

1. Mu-law encoders typically operate on 13-bit magnitude data, as opposed to 12-bit magnitude data with A-law.
2. Before chord determination a bias value of 33 is added to the absolute value of the linear input data to simplify the chord and step calculations.
3. The definition of sign bit is reversed, that is, input and output sign bit takes on the opposite sign.
4. The inversion pattern is applied to all bits in the 8-bit code.

The ***Table 34-13 Mu-law encoding*** illustrate Mu-law encoding algorithm. The sign bit S of the linear input data takes on the opposite value from the sign bit of the encode data.

**Table 34-13 Mu-law encoding**

| Linear Input Data | | | | | | | | | | | | | | Mu-law Encoded output | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | ~S | 0 | 0 | 0 | A | B | C | D |
| S | 0 | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | X | ~S | 0 | 0 | 1 | A | B | C | D |
| S | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | X | X | ~S | 0 | 1 | 0 | A | B | C | D |
| S | 0 | 0 | 0 | 0 | 1 | A | B | C | D | X | X | X | X | ~S | 0 | 1 | 1 | A | B | C | D |
| S | 0 | 0 | 0 | 1 | A | B | C | D | X | X | X | X | X | ~S | 1 | 0 | 0 | A | B | C | D |
| S | 0 | 0 | 1 | A | B | C | D | X | X | X | X | X | X | ~S | 1 | 0 | 1 | A | B | C | D |
| S | 0 | 1 | A | B | C | D | X | X | X | X | X | X | X | ~S | 1 | 1 | 0 | A | B | C | D |
| S | 1 | A | B | C | D | X | X | X | X | X | X | X | X | ~S | 1 | 1 | 1 | A | B | C | D |

After the input data is encoded through the algorithm defined in the chart, an inversion pattern is applied to the 8-bit code to increase the density of the transmission line, a benefit to the hardware performance. The inversion pattern is applied by XOR'ing the 8 bit code with 0xFF.

Decoding the Mu-law is essentially a matter of reversing the steps in the encoding. The ***Table***

*34-14 Mu-law decoding* illustrates the Mu-law decoding process, applied after reversing the inversion pattern. The least significant bits discarded in the encoding process are approximated by the median value of the interval. This is shown in the output section by the trailing 1…0 pattern after the D bit.

**Table 34-14 Mu-law decoding**

| Mu-law Encoded Input | | | | | | | | Linear Output Data | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S | 0 | 0 | 0 | A | B | C | D | ~S | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 |
| S | 0 | 0 | 1 | A | B | C | D | ~S | 0 | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 |
| S | 0 | 1 | 0 | A | B | C | D | ~S | 0 | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 |
| S | 0 | 1 | 1 | A | B | C | D | ~S | 0 | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 | 0 |
| S | 1 | 0 | 0 | A | B | C | D | ~S | 0 | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 | 0 | 0 |
| S | 1 | 0 | 1 | A | B | C | D | ~S | 0 | 0 | 1 | A | B | C | D | 1 | 0 | 0 | 0 | 0 | 0 |
| S | 1 | 1 | 0 | A | B | C | D | ~S | 0 | 1 | A | B | C | D | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| S | 1 | 1 | 1 | A | B | C | D | ~S | 1 | A | B | C | D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 34.3.15. Output drive

SAI can drive each audio sub-block's frame synchronization (FS), serial clock (SCK), and serial data (SD) independently of SAI enable status through the configuration of output drive (ODRIV) in SAI_BxCFG0 register.

Output drive settings must be programmed after SAI register configuration and before SAI is enabled.

### 34.3.16. IO management

IO management module is connected to both SAI audio sub-blocks, it is the only medium where they are connected. When audio sub-block is configured synchronous with the other sub-block through the synchronization mode (SYNCMOD) bit in SAI_BxCFG0 register, FS, SCK and MCLK pin could be shared, those pins of the synchronous sub-block are freed and left as general purpose IOs. When one audio sub-block is declared synchronous to the other audio block, it must be configure as slave.

This function is especially useful in duplex mode, where one sub-block act as transmitter, while the other act as receiver, the synchronous sub-block receive its FS and SCK signal through the IO management module, which comes from the asynchronous module if it is declared as master or from external IO if it is declared as slave.

### 34.3.17. DMA interface

Each SAI audio sub-block has its own DMA interface. DMA access is enable through DMA enable (DMAEN) bit in the SAI_BxCFG0 register. DMA request is generated together with FIFO request (FFREQ), whose status depend on FIFO threshold (FFTH) and FIFO status (FFSTAT), this is especially important when DMA burst transaction is used. When the audio sub-block is configured in transmitter mode, FIFO threshold must be set to a value such that

enough empty space is left for a complete DMA burst write operation in the worst case, otherwise FIFO overrun might occur. When the audio sub-block is configured in receiver mode, FIFO threshold must be set to a value such that enough data in the FIFO of a complete DMA burst read operation to avoid FIFO underrun condition.

DMA direction is linked to the audio sub-block operating configuration. When configured as transmitter, the DMA request a data load to the internal FIFO by writing the data register SAI_BxDATA. When configured as receiver, the DMA request a data read from the internal FIFO by reading the data register SAI_BxDATA.

**Note**: DAM SAI channel must be enabled after SAI register configuration.

### 34.3.18. Enable/Disable

SAI audio sub-block is enabled through the SAIEN bit in the SAI_BxCFG0 register, users must ensure this is done after the audio sub-block is configured, on-the-fly configuration is not supported, if done so, hardware's behavior is not guaranteed. Slave audio sub-block must be enabled before master audio block.

Users can disable the audio sub-block anywhere during an active frame transfer, but it will only be completely disabled at the end of current frame.

### 34.3.19. Error flags

#### Clock error configuration detection

Clock error configuration detection mechanism in enabled only when the audio sub-block is configured as master and clock divider bypass (BYPASS) is clear. In this operating mode, users have to guarantee that the frame length (FWD+1) equals the results of an exponential function of base 2 within 8- and 256-bits range, otherwise the clock error flag (ERRCK) will be set in the status register SAI_BxSTAT. The frame length must be a power of 2 is to ensure there is an integer number of master clock (MCLK) cycles within each bit clock cycle (SCK), for better sound quality.

An interrupt is generated if the clock error configuration detection interrupt enable (ERRCKIE) bit in the interrupt enable register SAI_BxINTEN is set.

When Clock error is detected, it automatically disables the audio sub-block via clearing the SAIEN in the SAI_BxCFG0 register.

#### Audio codec not ready detection

Audio codec not ready status is checked only when AC'97 protocol is used and receiver operating condition is choose. The audio sub-block determines audio codec ready status form the TAG (the first slot). When bit 15 of the TAG is 0, audio codec not ready flag (ACNRDY) in the status register SAI_BxSTAT is set, an interrupt is generated if audio codec not ready interrupt enable (ACNRDYIE) in the interrupt enable register SAI_BxINTEN is set. When

codec not ready is detected, data contained in the following slots of the current frame will not be loaded into the FIFO.

Audio codec not ready detected flag is clear by setting the audio codec not ready detected clear (ACNRDYC) bit in the SAI_BxINTC register.

### Frame synchronization advanced detection

Frame synchronization advanced detection mechanism is enabled only when the audio sub-block is configured as a slave, since salve receives the FS signal, and its arrival time is crucial to correct data interpretation. Frame synchronization advanced detection is possible because frame length, frame active polarity and frame offset are determine before audio sub-block enabled.

Frame synchronization advancement has no effect on the current frame since FS active edge is only anticipated at the end of the frame.

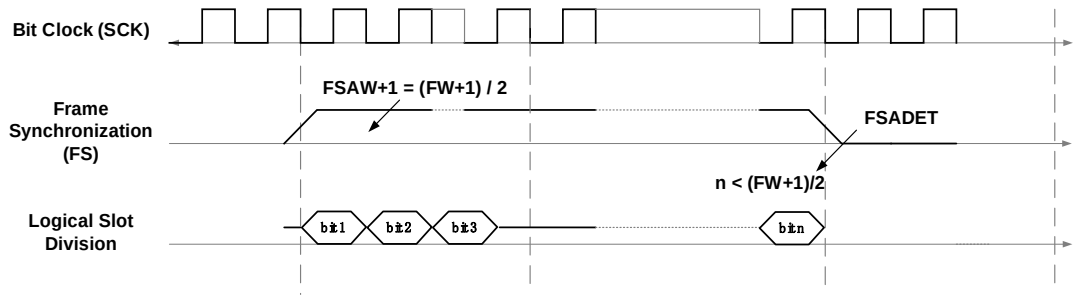
An interrupt is generated when both frame synchronization advanced detected flag (FSADET) in status register SAI_BxSTAT and frame synchronization advanced detection interrupt enable in interrupt enable SAI_BxINTEN register are set.

To resynchronize, the following steps are needed:

1. Audio sub-block should be disabled, users must wait for the SAIEN control field of the corresponding sub-block is completely disabled.
2. The internal FIFO should be flushed by setting the FLUSH control field.
3. Enable the audio sub-block again by setting SAIEN.
4. Wait for FS to restart synchronization.

**Note**: This flat is not generated in AC'97 configuration mode. Since AC'97 is only a link controller, FS is generated even if the audio sub-block is configured as a slave.

**Figure 34-24 Frame synchronization advanced detection**




### Frame synchronization postponed detection

Frame synchronization postponed detected mechanism is enabled only when the audio sub-block is configured as a slave, since the salve receives the FS signal, and its arrival time is crucial to correct data interpretation. Frame synchronization postpone detection is possible because frame length, frame active polarity and frame offset are determined before audio sub-block is enabled.

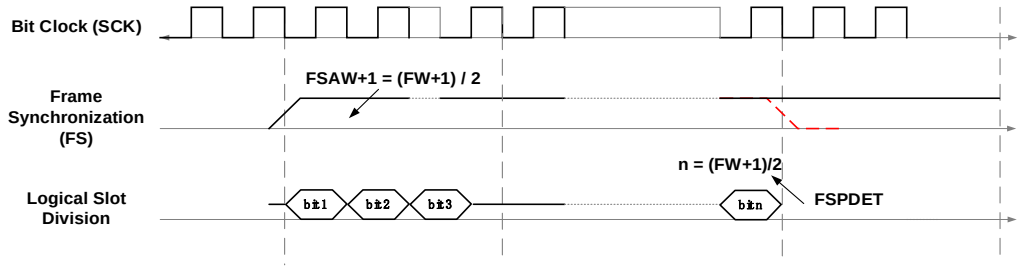
Frame synchronization postpone might be caused by late generation of the master device, external delay or noise induced glitches. This incorrect FS timing could disrupt audio sub-block's internal finite state machine, thus corrupting correct transactions.

An interrupt is generated when both frame synchronization postpone detected flag (FSPDET) in status register SAI_BxSTAT and frame synchronization postponed interrupt enable in interrupt enable SAI_BxINTEN register are set.

To resynchronize with the master, the steps for resynchronization should be applied.

**Note**: This flat is not generated in AC'97 configuration mode. Since AC'97 is only a link controller, FS is generated even if the audio sub-block is configured as a slave.

**Figure 34-25 Frame synchronization postponed detection**




## FIFO overrun or underrun detection

FIFO overrun and underrun flag (OUERR) ccupies the same bit in status register SAI_BxSTAT since each audio sub-block can be configured into either transmitter or receiver.

When the audio sub-block is configures as transmitter, underrun can happen during an active frame transaction when FIFO is empty and a new slot of empty data is sent. An interrupt is generated if the overrun or underrun interrupt enable (OUERRIE) in the interrupt enable register SAI_BxINTEN is set. If underrun happens, a re-synchronization procedure is showed in the following steps:

1. Audio sub-block should be disabled, users must wait until SAIEN control field of the corresponding sub-block is completely disabled.
2. The internal FIFO should be flushed by setting the FLUSH control field.
3. Fill the correct data to be transferred in into the FIFO.
4. Enable the audio sub-block again by setting SAIEN.

Underrun flag is cleared by setting the overrun or underrun clear (OUERRC) bit in the interrupt clear register SAI_BxINTC.

When the audio sub-block is configured as receiver, overrun can happen during an active frame transaction when FIFO is full and a new slot of data is received. When overrun happens, the newly received data is discarded, nothing will be written into the FIFO. An interrupt is generated if the overrun or underrun interrupt enable (OUERRIE) in the interrupt enable register SAI_BxINTEN is set.

Overrun flag is also cleared by setting the OUERRC bit in the SAI_BxINTC register.

**Note**: When DMA is enabled, users have to guarantee correct DMA configuration, especially if DMA burst is used, otherwise overrun and underrun can both happen in transmitter or receiver operating mode.

### 34.3.20. Interrupts

The ***Table 34-15 Interrupt control*** summarizes all the interrupt sources present in each audio sub-block.

**Table 34-15 Interrupt control**

| Interrupt source | Interrupt partition | Interrupt raise condition | Interrupt Enable Control | Interrupt Clear Control |
|---|---|---|---|---|
| FFREQ | Request | OPTMOD = Any | FFREQIE | Read/Write SAI_BxDATA |
| MTDET | Mute | OPTMOD = Receiver | MTDETIE | MTDETC |
| ERRCK | Error | OPTMOD = Master; BYPASS = 1 | ERRCKIE | ERRCKC |
| ACNRDY | Error | OPTMOD = Slave; PROT = AC'97 | ACNRDYIE | ACNRDYC |
| FSADET | Error | OPTMOD = Slave; PROT ≠AC'97 | FSADETIE | FSADETC |
| FSPDET | Error | OPTMOD = Slave; PROT ≠AC'97 | FSPDETIE | FSPDETC |
| OUERR | Error | OPTMOD = Any | OUERRIE | OUERRC |

To recover from error interrupt, the following procedure should be applied:

1. Disable the corresponding interrupt.
2. Configure SAI function registers.
3. Enable interrupt.
4. Enable SAI audio sub-block.

## 34.4. Register definition

SAI0 base address: 0x4001 5800

SAI1 base address: 0x4001 5C00

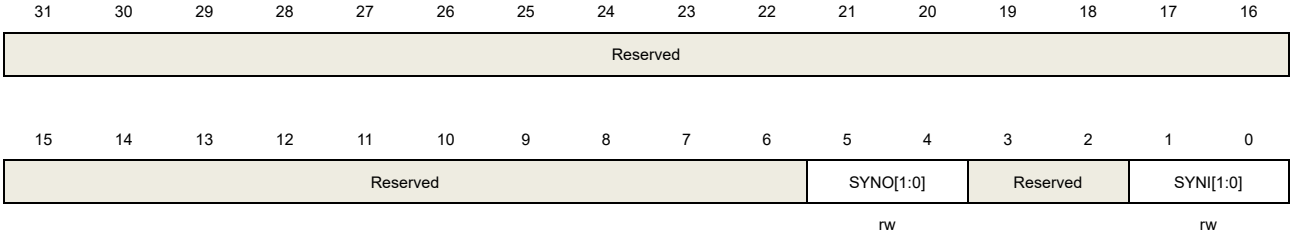
SAI2 base address: 0x4001 6000

### 34.4.1. Synchronize configuration register (SAI_SYNCFG)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



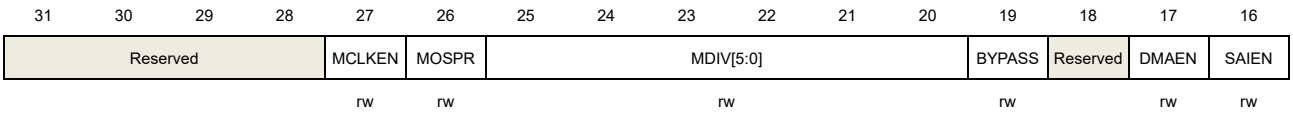

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:4 | SYNO[1:0] | Synchronization outputs<br>00: No synchronization output signals.<br>01: Block 0 used for further synchronization for others SAI<br>10: Block 1 used for further synchronization for others SAI<br>11: Reserved. These bits must be set when both audio block (0 and 1) are disabled.<br>**Note:** SYNO[1:0] should be configured as No synchronization output signals when audio block is configured as SPDIF. |
| 3:2 | Reserved | Must be kept at reset value. |
| 1:0 | SYNI[1:0] | Synchronization inputs.<br>Refer to ***Table 34-2. External synchronization configuration***.<br>They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCMOD[1:0] = 10 in SAI_BxCFG0). |

### 34.4.2. Block x configuration register0 (SAI_BxCFG0) (x = 0,1)

Address offset: 0x04 + 0x20 * x

Reset value: 0x0000 0040

This register has to be accessed by word (32-bit).



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ODRIV | MONO | SYNCMOD[1:0] | | SAMPEDGE | SHIFTDIR | DATAWD[2:0] | | | Reserved | PROT[1:0] | | OPTMOD[1:0] | |
| | | rw | rw | rw | | rw | rw | rw | | | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27 | MCLKEN | The master clock enable<br>0: The master clock is enable<br>1: The master clock is enabled independently of SAIEN bit |
| 26 | MOSPR | The master clock oversampling rate.<br>0: MCLK = 256 * $F_{fs}$<br>1: MCLK = 512 *$F_{fs}$ |
| 25:20 | MDIV[5:0] | Master clock divider ratio.<br>0000: Primary frequency divider logic bypass,<br>Otherwise, please refer to formula 30-1 in chapter ***Clock divider*** for output frequency calculation by the formula<br>**Note**: This control field is ineffective when SAI configured in slave mode.<br>**Note**: This control field has to be set before SAI is enabled. |
| 19 | BYPASS | Clock divider logic bypass.<br>0: Clock divider ratio is applied to both primary and secondary divider logic.<br>1: Clock divider logic is bypassed. |
| 18 | Reserved | Must be kept at reset value. |
| 17 | DMAEN | DMA enable.<br>0: DMA disabled<br>1: DMA enabled<br>**Note**: DMAEN must be set after OPTMOD control field if SAI is configured as receiver to avoid unnecessary DMA request, since SAI is transmitter after reset. |
| 16 | SAIEN | SAI sub-block enable.<br>0: SAI sub-block is disabled.<br>1: SAI sub-block is enabled, this control field can only be set when SAIEN is 0.<br>**Note**: The SAI sub-block is completely disabled (SAIEN read as 0) only at the end of the current frame if software has already issued a disabled request somewhere during the on-going transfer.<br>**Note**: SAI slave must be enabled before SAI master. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | ODRIV | Output Drive.<br>0: SAI sub-block output driven only when SAIEN is set.<br>1: SAI sub-block output driven according to ODRIV setting.<br>**Note**: This control field has to be set after SAI configuration but before SAI sub- |

| | | |
|---|---|---|
| | | block enabled. |
| 12 | MONO | Stereo and Mono mode selection.<br>0: Stereo mode.<br>1: Mono mode.<br>Mono mode requires slot number equals to 2, in transmitter mode, the first slots data is copied to the second slot, while in receiver mode, the second slot's data is ignored. |
| 11:10 | SYNCMOD[1:0] | Synchronization mode.<br>00: Asynchronous with the other sub-block.<br>01: Synchronous with the other sub-block, when this mode is selected, users must configure the operating mode to slave.<br>10: Synchronous with an external SAI audio sub-block, when this mode is selected, users must configure the operating mode to slave.<br>11: Reserved.<br>**Note**: It is required to be configured when audio sub-block is disabled.<br>**Note**: If the protocol select as SPDIF, the mode should be configured as asynchronous. |
| 9 | SAMPEDGE | Sampling clock edge.<br>0: Data sampled on SCK falling edge.<br>1: Data sampled on SCK rising edge.<br>**Note**: This control field is ignored in SPDIF mode.<br>**Note**: It is required to be configured when audio sub-block is disabled. |
| 8 | SHIFTDIR | Shift direction.<br>0: Data is shifted with MSB first.<br>1: Data is shifted with LSB first.<br>**Note**: This control field is ignored in AC'97 mode, since shift direction is forced to MSB first. |
| 7:5 | DATAWD[2:0] | Data width.<br>000: Reserved.<br>001: Reserved.<br>010: 8-bits wide<br>011: 10-bits wide.<br>100: 16-bits wide.<br>101: 20-bits wide.<br>110: 24-bits wide.<br>111: 32-bits wide.<br>In compander mode, data width is fix to 8-bits width by the algorithm itself.<br>**Note**: It is required to be configured when audio sub-block is disabled.<br>**Note**: This control field is ignored in SPDIF mode.<br>**Note**: In case AC'97 protocol is selected, only 16- or 20-bits is viable, otherwise audio sub-block's behavior is not guaranteed. |

| | | |
|---|---|---|
| 4 | Reserved | Must be kept at reset value. |
| 3:2 | PROT[1:0] | Protocol selection.<br>00: Polymorphic.<br>01: SPDIF.<br>10: AC'97.<br>10: Reserved.<br>Polymorphic configuration allows the user to tweak with all the frame and slot configuration options to form his protocol of choice suck as I2S, LSB/MSB justified, TDM, PCM/DSP and so on.<br>**Note**: It is required to be configured when audio sub-block is disabled. |
| 1:0 | OPTMOD[1:0] | Operating mode.<br>00: Master transmitter.<br>01: Master receiver.<br>10: Slave transmitter.<br>11: Slave receiver.<br>If the protocol select as SPDIF, the operating mode will be forced to configure as master transmitter.<br>**Note**: It is required to be configured when audio sub-block is disabled. |

### 34.4.3. Block x configuration register1 (SAI_BxCFG1) (x = 0,1)

Address offset: 0x08 + 0x20 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPAMOD[1:0] | | CPLMOD | MTFCNT[5:0] | | | | | | MTVAL | MT | SDOM | FLUSH | FFTH[2:0] | | |
| rw | | rw | rw | | | | | | rw | rw | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:14 | CPAMOD[1:0] | Compander mode.<br>00: No compansion applied.<br>01: Reserved.<br>10: Mu-law algorithm.<br>11: A-law algorithm.<br>ITU-T G.711 defines two main compansion algorithm, the Mu-law and A-law, which encode 13- and 12-bits signed linear PCM respectively to logarithmic 8-bit samples. The former gives more resolution to higher range signals while the later provides more quantization levels at lower signal levels. |

| | | |
|---|---|---|
| | | The compression or expansion mode is selected via control field OPTMOD[0], when the audio sub-block is configured as transmitter, compression is automatically applied, while expansion is performed when configured as receiver. The complement mode is selected via control field CPLMOD.<br>**Note**: Compansion should be applied when TDM protocol is configured. |
| 13 | CPLMOD | Complement mode.<br>This control field is used to select the complement option in compansion mode.<br>0: Data represented in 1's complement form.<br>1: Data represented in 2's complement form. |
| 12:7 | MTFCNT[5:0] | Mute frame count.<br>This control field is only used in reception mode. When the number of consecutive mute frame received equals to MTFCNT, MTFDET flag is set and an interrupt is generated if MTFDETIE is set. |
| 6 | MTVAL | Mute value.<br>0: 0 is sent via the Serial Data (SD) line when mute is on.<br>1: If SLOTNUM is less or equals to two, last frame is sent via the Serial Data (SD) line when mute is on, Otherwise 0 is sent via the Serial Data (SD) line in mute frames.<br>**Note**: This control field is meaningful only when the audio sub-block is configured as transmitter, and is ignored in SPDIF mode.<br>**Note**: The receiver can only detect 0 valued mute frame, when MTVAL set to 1 and the last frame is repeatedly sent during the mute, the receiver will receive the data as it is, the mute frame counter will not amount, and so no MTDET flag will ever be set. |
| 5 | MT | Mute.<br>0: Mute mode off.<br>1: Mute mode on.<br>Note: This control field is meaningful only when the audio sub-block is configured as transmitter, SD output is determined by MTVAL when Mute is on.<br>Note: Mute will take effect at next frame if mute is set somewhere inside the on-going frame. |
| 4 | SDOM | Serial data output management.<br>0: SD output is driven entirely during the audio frame.<br>1: SD output is released near inactive slots.<br>**Note**: If the data offset of the first slot of the first frame is not equals to zero, SD will remain released until the first active data bit. If the current frame is not the first frame of a continuous transfer, whether the offset section of the first slot is released or not depends on the last slot of the previous frame. If the last slot is active, then the offset section is driven, otherwise it is released.<br>**Note**: If data offset plus data width is still lower than the slot width, than the space between the last bit of data and the end of slot is known as the empty section. SD output could be released during the empty section depends on whether the |

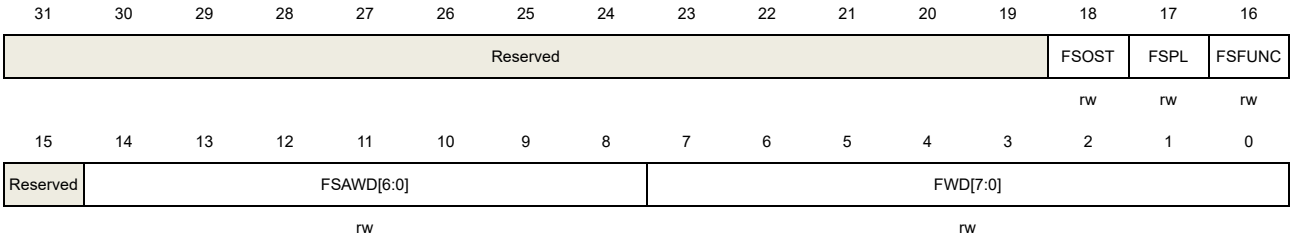
| | | |
|---|---|---|
| | | following slot is active, if the current slot is the last slot, then its empty section behavior depends on the first slot of current frame, it is independent on whether the current frame is the last frame.<br>**Note**: Whether SD outputs is driven or not on the empty section precede a slot and the offset section followed by a slot is determined by that slot only, if that slot is active, the SD line will be driven, otherwise it will be released. |
| 3 | FLUSH | FIFO flush.<br>0: No FIFO flush.<br>1: Perform FIFO flush.<br>**Note**: FIFO flush clears all data content inside the FIFO and reset read/write pointers. FIFO flush should be performed when SAI is disabled. |
| 2:0 | FFTH[2:0] | FIFO threshold.<br>000: FIFO empty.<br>001: FIFO quarter full.<br>010: FIFO half full.<br>011: FIFO three quarter full.<br>100: FIFO full.<br>101: Reserved.<br>110: Reserved.<br>111: Reserved.<br>**Note**: FIFO threshold is used in conjunction with FIFO status (FFSTAT) control field to determine FIFO request (FFREQ) is CPU and DMA mode. |

### 34.4.4. Block x frame configuration register (SAI_BxFCFG) (x = 0,1)

Address offset: 0x0C + 0x20 * x

Reset value: 0x0000 0007

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18 | FSOST | Frame synchronization offset.<br>0: FS active edge asserted at the beginning of the first bit of the first slot.<br>1: FS active edge asserted one bit cycle before normal FS when FSOST is 0.<br>**Note**: This control field must be configured before the audio sub-block is enabled, |

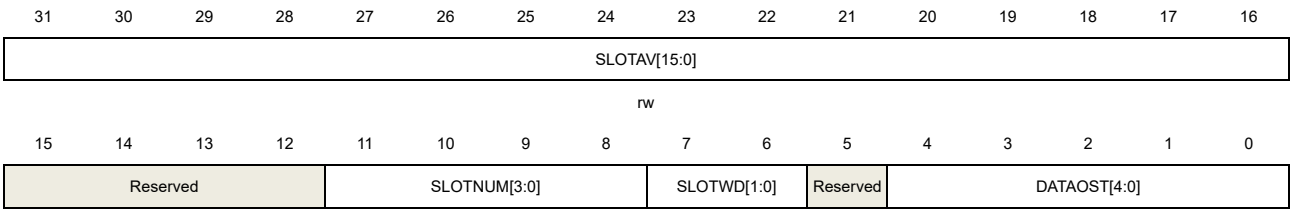
| | | |
|---|---|---|
| | | and it is meaningless when protocol is set to AC'97 or SPDIF. |
| 17 | FSPL | Frame synchronization active polarity.<br>0: FS low active polarity<br>1: FS high active polarity.<br>**Note**: This control field must be configured before the audio sub-block is enabled, and it is meaningless when protocol is set to AC'97 or SPDIF. |
| 16 | FSFUNC | Frame synchronization function.<br>0: FS only defines frame start.<br>1: FS define both frame start and channel number.<br>**Note**: This control field must be configured before the audio sub-block is enabled.<br>**Note**: When FSFUNC is set, slot number (SLOTNUM + 1) within a frame must be even, in this case half of the slot will be allocated to channel A, the other half to channel B. If the combination of all slots allocated to one channel is smaller than half the frame width, SD output will be release where slot is undefined, independent of SDOM.<br>**Note**: When FSFUNC is set, FS active width (FSAWD + 1) must be configure as half the length of a frame. |
| 15 | Reserved | Must be kept at reset value. |
| 14:8 | FSAWD[6:0] | Frame synchronization active width.<br>**Note**: This control field must be configured before the audio sub-block is enabled, and it is meaningless in AC'97 mode.<br>**Note**: This control field specified the active width of FS in terms of (FSAWD + 1) SCK clock cycles. |
| 7:0 | FWD[7:0] | Frame width<br>**Note**: This control field must be configured before the audio sub-block is enabled, and it is meaningless in AC'97 mode.<br>**Note**: This control field specified the frame width in terms of (FWD + 1) SCK clock cycles, When the audio sub-block is configured in master mode, and BYPASS = 0, (FWD+1) must equals the results of an exponential function of base 2 within 8- and 256-bits range. |

### 34.4.5. Block x slot configuration register (SAI_BxSCFG) (x = 0,1)

Address offset: 0x10 + 0x20 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | SLOTAV[15:0] | Slot activation vector.<br>0: Slot inactive.<br>1: Slot active.<br>Each bit in the SLOTAV vector is aligned to the slot number from 0 to 15, if SLOTNUM is less than 15, the unaligned vector bits is ignored.<br>**Note**: This control field must be configured before the audio sub-block is enabled.<br>**Note**: This control field is meaningless in AC'97 or SPDIF mode. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:8 | SLOTNUM[3:0] | Slot number within frame.<br>The actual slot number with in a frame is (SLOTNUM + 1), and cannot exceed 16. When FSFUNC is set, slot number must be even for equals' allocation to both channels.<br>**Note**: This control field must be configured before the audio sub-block is enabled.<br>**Note**: This control field is meaningless in AC'97 mode. |
| 7:6 | SLOTWD[1:0] | Slot width.<br>00: Slot width equals' data width.<br>01: Slot width of 16-bits.<br>10: Slot width of 32-bits.<br>11: Reserved.<br>Slot width must be greater or equal to data width to hold the data, Otherwise the behavior is not guaranteed.<br>**Note**: This control field must be configured before the audio sub-block is enabled.<br>**Note**: This control field is meaningless in AC'97 or SPDIF mode. |
| 5 | Reserved | Must be kept at reset value. |
| 4:0 | DATAOST[4:0] | Data offset.<br>Defines when the first bit of data appear within an active slot, in transition mode, the SD output in the offset and empty section depends on SDOM and nearby slot activation status, 0 or Hi-Z will be outputted. While in reception mode, data content on the offset and empty section will be ignored.<br>**Note**: This control field must be configured before the audio sub-block is enabled.<br>**Note**: This control field is meaningless in AC'97 mode. |

### 34.4.6. Block x interrupt enable register (SAI_BxINTEN) (x = 0,1)

Address offset: 0x14 + 0x20 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FSPDETIE | FSADETIE | ACNRDYIE | FFREQIE | ERRCKIE | MTDETIE | OUERRIE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | FSPDETIE | Frame synchronization postpone detection interrupt enable.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if FSPDET and FSPDETIE are both set.<br>**Note**: This control field is meaningless when the audio sub-block is configured as master.<br>**Note**: This control field is meaningless in AC'97 mode. |
| 5 | FSADETIE | Frame synchronization advanced detection interrupt enable.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if FSADET and FSADETIE are both set.<br>**Note**: This control field is meaningless when the audio sub-block is configured as master.<br>**Note**: This control field is meaningless in AC'97 mode. |
| 4 | ACNRDYIE | Audio codec not ready interrupt enable.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if ACNRDY and ACNRDYIE are both set.<br>**Note**: This control field has a meaning only when the audio sub-block is configured as receiver.<br>**Note**: This control field has a meaning only when AC'97 mode is selected. |
| 3 | FFREQIE | FIFO request interrupt enable.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if FFREQ and FFREQIE are both set.<br>**Note**: When the audio sub-block is configured as receiver, OPTMOD must be set before FFREQIE is enabled to guarantee no false FIFO request is generated since the sub-block is transmitter after reset. |
| 2 | ERRCKIE | Error clock interrupt enable. It is set and cleared by software.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if ERRCK and ERRCKIE are both set.<br>**Note**: This control field is relevant only if the audio sub-block is configured as master |

| | | |
|---|---|---|
| | | and the BYPASS is set to 0 in the clock divide logic.<br>**Note**: This control filed is used when TDM mode is selected, it is meaningless in other modes. |
| 1 | MTDETIE | Mute detection interrupt enable.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if MTDET and MTDETIE are both set.<br>Note: this control field has a meaning only when the audio sub-block is configured as receiver. |
| 0 | OUERRIE | FIFO overrun or underrun interrupt enable.<br>0: Interrupt disabled.<br>1: Interrupt enabled.<br>An interrupt is generated if OUERR and OUERRIE are both set. |

### 34.4.7. Block x status register (SAI_BxSTAT) (x = 0,1)

Address offset: 0x18 + 0x20 * x

Reset value: 0x0000 0008

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | FFSTAT[2:0] | | |
| | | | | | | | | | | | | | | r | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FSPDET | FSADET | ACNRDY | FFREQ | ERRCK | MTDET | OUERR |
| | | | | | | | | | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18:16 | FFSTAT[2:0] | FIFO status.<br>Indicate the full/empty status of the FIFO, and it is controlled solely by hardware, where different evaluation standards is present according to audio sub-blocks operating mode.<br>In case OPTMOD is configured as receiver:<br>000: Empty.<br>001: Empty <FIFO_Level<= 1/4_Full.<br>010: 1/4_Full <FIFO_Level<= 1/2_Full.<br>011: 1/2_Full <FIFO_Level<= 3/4_Full.<br>100: 3/4_Full <FIFO_Level< Full<br>101: Full.<br>And when OPTMOD is configured as transmitter:<br>000: Empty.<br>001: Empty <FIFO_Level< 1/4_Full. |

| | | |
|---|---|---|
| | | 010: 1/4_Full <= FIFO_Level< 1/2_Full.<br>011: 1/2_Full <= FIFO_Level< 3/4_Full.<br>100: 3/4_Full <= FIFO_Level< Full<br>101: Full. |
| 15:7 | Reserved | Must be kept at reset value. |
| 6 | FSPDET | Frame synchronization postponed detection.<br>0: Correct FS edge reception.<br>1: FS edge reception postponed.<br>FS edge reception postponed could generate an interrupt if FSPDETIE is set. This flag is cleared by FSPDETC control filed.<br>**Note**: This control field has a meaning only when the audio sub-block is configured as slave. |
| 5 | FSADET | Frame synchronization advanced detection.<br>0: Correct FS edge reception.<br>1: FS edge reception advanced.<br>FS edge reception advanced could generate an interrupt if FSADETIE is set. This flag is cleared by FSADETC control filed.<br>**Note**: This control field has a meaning only when the audio sub-block is configured as slave. |
| 4 | ACNRDY | Audio codec not ready.<br>0: AC'97 codec ready.<br>1: AC'97 codec not ready.<br>Bit-15 in TAG slot of each frame is the AC'97 audio codec ready indication bit. 0 indicate the audio codec is not ready, on the other hand, 1 means ready.<br>AC'97 codec not ready could generate an interrupt if ACNRDYIE is set. This flag is cleared by ACNRDYC control field.<br>**Note**: This control field is used only in AC'97 mode. |
| 3 | FFREQ | FIFO request<br>0: No FIFO request.<br>1: FIFO write or read request.<br>FIFO request could generate an interrupt if FFREQIE is set.<br>The request type depend on audio sub-block configuration, if OPTMOD is configured as transmitter and all conditions met, write request in generated, else if receiver mode is selected, read request is generated. |
| 2 | ERRCK | Error clock.<br>0: Correct clock configuration<br>1: Error clock configuration.<br>Error clock configuration could generate an interrupt if ERRCKIE is set. This flag is cleared by ERRCKC control field.<br>This control field has a meaning only when the audio sub-block is configured as master and BYPASS is set to 0. |

| | | |
|---|---|---|
| 1 | MTDET | Mute detection.<br>0: No mute detected<br>1: Mute detected.<br>Mute detection could generate an interrupt if MTDETIE is set. This flag is cleared by MTDETC control field.<br>When the number of frame whose slots received is all 0 value reaches the frame number defined in MTFCNT, mute detection flag is set. Mute could not be detected when slot number is less the 2 and MTVAL is set, in the transmitter, where the frame before mute is transferred repeatedly. |
| 0 | OUERR | Overrun or underrun.<br>0: No FIFO overrun or underrun detected<br>1: FIFO overrun or underrun detected.<br>FIFO overrun or underrun interrupt could be generated if OUERRIE is set. This flag is cleared by OUERRC control field.<br>FIFO overrun is generated only when the audio sub-block is configured as receiver, the received data fill the FIFO and the FIFO is full.<br>FIFO underrun is generated only when the audio sub-block is configured as transmitter, the FIFO is empty and a transfer request is present. |

### 34.4.8. Block x interrupt flag clear register (SAI_BxINTC) (x = 0,1)

Address offset: 0x1C + 0x20 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FSPDETC | FSADETC | ACNRDYC | Reserved | ERRCKC | MTDETC | OUERRC |
| | | | | | | | | | rw | rw | rw | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | FSPDETC | Frame synchronization postponed detection interrupt clear.<br>Writing 1 clears FSPDET flag.<br>**Note**: This control field is not used in AC'97 or SPDIF mode.<br>**Note**: This field always reads as 0. |
| 5 | FSADETC | Frame synchronization advanced detection interrupt clear.<br>Writing 1 clears FSADET flag.<br>**Note**: This control field is not used in AC'97 or SPDIFmode. |

| | | |
|---|---|---|
| | | **Note**: This field always reads as 0. |
| 4 | ACNRDYC | Audio codec not ready interrupt clear.<br>Writing 1 clears ACNRDY flag.<br>**Note**: This control field is only used in AC'97 mode.<br>**Note**: This field always reads as 0. |
| 3 | Reserved | Must be kept at reset value. |
| 2 | ERRCKC | Clock error interrupt clear.<br>Writing 1 clears ERRCK flag.<br>**Note**: This control field is only used when the audio sub-block is configured as master and BYPASS is set to 0.<br>Note: This field always reads as 0. |
| 1 | MTDETC | Mute detection interrupt clear.<br>Writing 1 clears MTDET flag.<br>**Note**: This field always reads as 0. |
| 0 | OUERRC | Overrun or underrun interrupt clear.<br>Writing 1 clears OUERR flag.<br>**Note**: This field always reads as 0. |

### 34.4.9. Block x data register (SAI_BxDATA) (x = 0,1)

Address offset: 0x20 + 0x20 * x

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DATA[31:0] | Data.<br>Write and read operations are performed on the FIFO directly. |

### 34.4.10. PDM control register (SAI_PDMCTL)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved |
|---|

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | CLKL1EN | CLKL0EN | Reserved | | MICNUMSEL | | Reserved | | | PDMEN |
| | | | | | | rw | rw | | | rw | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | CLKL1EN | PDM clock line 1 enable.<br>0: PDM clock line 1 disable<br>1: PDM clock line 1 enable |
| 8 | CLKL0EN | PDM clock line 0 enable.<br>0: PDM clock line 0 disable<br>1: PDM clock line 0 enable |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | MICNUMSEL | Select microphones number<br>00: 2 microphones<br>01: 4 microphones<br>10: 6 microphones<br>11: 8 microphones |
| 3:1 | Reserved | Must be kept at reset value. |
| 0 | PDMEN | PDM enable<br>0: PDM disable<br>1: PDM enbale |

### 34.4.11. PDM configuration register (SAI_PDMCFG)

Address offset: 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | DPR3[2:0] | | | Reserved | DPL3[2:0] | | | Reserved | DPR2[2:0] | | | Reserved | DPL2[2:0] | | |
| | rw | | | | rw | | | | rw | | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | DPR1[2:0] | | | Reserved | DPL1[2:0] | | | Reserved | DPR0[2:0] | | | Reserved | DPL0[2:0] | | |
| | rw | | | | rw | | | | rw | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:28 | DPR3[2:0] | The 3rd group of right channel microphone data flow delay period |

| | | |
|---|---|---|
| | | 000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |
| 27 | Reserved | Must be kept at reset value. |
| 26:24 | DPL3[2:0] | The 3rd group of left channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |
| 23 | Reserved | Must be kept at reset value. |
| 22:20 | DPR2[2:0] | The 2nd group of right channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | DPL2[2:0] | The 2nd group of left channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | DPR1[2:0] | The 1st group of right channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |
| 11 | Reserved | Must be kept at reset value. |
| 10:8 | DPL1[2:0] | The 1st group of left channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>… |

| | | |
|---|---|---|
| | | 111: delay 7 $T_{SAI_CK}$ periods |
| 7 | Reserved | Must be kept at reset value. |
| 6:4 | DPR0[2:0] | The 0 group of right channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | DPL0[2:0] | The 0 group of left channel microphone data flow delay period<br>000: No delay<br>001: delay 1 $T_{SAI_CK}$ period<br>010: delay 2 $T_{SAI_CK}$ periods<br>…<br>111: delay 7 $T_{SAI_CK}$ periods |

# 35. Image processing accelerator (IPA) (For GD32H737_757_759)

## 35.1. Overview

The IPA provides a configurable and flexible image format conversion from one or two source image to the destination image, with the following four conversion modes:

- Copy one source image to the destination image
- Convert one source image to the destination image with specific pixel format
- Convert and blend two source images to the destination image with specific pixel format
- Fill up the destination image with a specific color

Sixteen pixel formats for foreground from 4-bit up to 32-bit per pixel, eleven pixel formats for background from 4-bit up to 32-bit per pixel, and five pixel formats from 16-bit up to 32-bit per pixel for the destination image are supported. Two 256*32 bits LUTs (Look-Up Table) separately for the two source images are implemented for the indirect pixel formats.

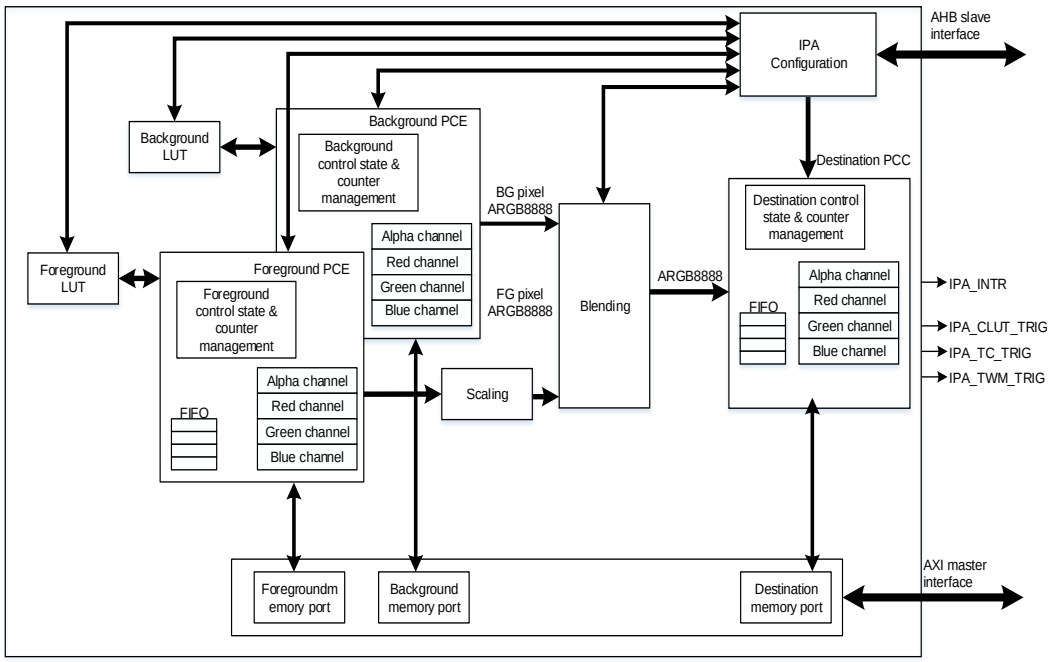
## 35.2. Characteristics

- One AXI master interface for memory access
- One AHB slave interface for IPA configuration with 8-bit, 16-bit, 32-bit
- Three four-doubleword depth 64-bit FIFOs independently for the source and destination images
- Support four pixel-format-convert modes
  - Copy one source image to the destination image
  - Convert one source image to the destination image with specific pixel format
  - Convert and blend two source images to the destination image with specific pixel format
  - Fill up the destination image with a specific color
- Support configurable LUT size independently for two source images
- Support two LUT pixel formats separately for two source images
- Support LUT automatically loading for two source images
- Support transfer hang up and stop
- Support pixel offset per line independently for the source and destination images
- Support pre-defined pixel channel value independently for the source and destination images
- Support three alpha channel value calculation algorithms separately for two source images
- Support sixteen pixel formats for foreground image
- Support eleven pixel formats for background image
- Support five pixel formats for the destination image

- Support configurable image size
- Support automatically AXI bandwidth adjustment with an internal timer
- Support one interrupt with six types of event flags
- Support interrupt enable and clear
- Support decimal scaling and bilinear scaling
- Support image rotation (0, 90, 180, 270 degrees)
- Support interlace input for the foreground image

## 35.3. Block diagram

**Figure 35-1. IPA block diagram**




As showed in ***Figure 35-1. IPA block diagram***, the IPA consists of 7 main parts:

- IPA configuration through AHB slave interface
- Image data access through AXI master interface
- Foreground and background LUT
- Foreground and background pixel channel extension (PCE)
- Foreground scaling block
- Foreground and background pixel blending
- Destination pixel channel compression (PCC)

## 35.4. Signal description

The signals of IPA are listed in ***Table 35-1. Signals description of IPA***.

**Table 35-1. Signals description of IPA**

| I/O port | Type | Description |
|---|---|---|
| IPA_INTR | O | IPA global interrupt signal |
| IPA_CLUT_TRIG | O | IPA CLUT transfer complete trigger signal |
| IPA_TC_TRIG | O | IPA transfer complete trigger signal |
| IPA_TWM_TRIG | O | IPA transfer watermark trigger signal |

## 35.5. Function overview

The IPA is a pixel format converter, supporting multiple conversion modes, foreground pixel formats and line offset, background pixel formats and line offset, destination pixel formats and line offset to allow for flexible application by configuring the corresponding bits in the IPA registers. All the IPA registers (expect for LUT accesses only 32-bit supported) can be 8-bit, 16-bit and 32-bit configured through AHB slave interface.

Four conversion modes are supported, which is determined by the PFCM bits in the IPA_CTL register, as listed in the ***Table 35-2. IPA conversion mode***.

- Copy foreground image to the destination image

  In this mode, the pixel data in the foreground memory are copied to the destination memory without pixel conversion. So the configured pixel format of the foreground and destination images have no specific meaning. The foreground pixel format only defines the bit number per pixel.

- Convert foreground image to the destination image

  In this mode, the pixel data in the foreground memory are converted from the foreground pixel format to the destination pixel format, and then written into the destination memory. If the foreground pixel format is indirect (L8, AL44, AL88, L4), the data read from the foreground memory is used as an index to retrieve the pixel data from the foreground LUT.

- Convert and blend the foreground and background images to the destination image

  In this mode, the pixel data in the foreground and background memory are firstly converted from the foreground and background pixel format to ‘ARGB8888’. Pairs of foreground and background pixel value are blended and converted from ‘ARGB8888’ to the destination pixel format, and then written into the destination memory.

  If the foreground pixel format is indirect, the data read from the foreground memory is used as an index to retrieve the pixel data from the foreground LUT.

  If the background pixel format is indirect, the data read from the background memory is used as an index to retrieve the pixel data from the background LUT.

- Fill up the destination image with a specific color

  In this mode, the destination image is filled up with the pre-defined pixel channel value,

corresponding with the destination pixel format.

**Table 35-2. IPA conversion mode**

| PFCM[1:0] | Conversion mode | | Pixel conversion | Blending |
|---|---|---|---|---|
| | Source | Destination | | |
| 00 | Foreground image | Destination image | No | No |
| 01 | Foreground image | Destination image | Yes | No |
| 10 | Foreground and background image | Destination image | Yes | Yes |
| 11 | Pixel value pre-defined in the register | Destination image | No | No |

### 35.5.1. Conversion operation

An IPA transaction consists of seven operations:

1) Read pixel data from the foreground memory addressed through the IPA_FMADDR. Retrieve the pixel data from the foreground LUT if the foreground pixel format is indirect.
2) Extend the foreground pixel value to a 32-bit value, and calculate the alpha channel value according to the FAVCA bits in the IPA_FPCTL register.
3) Read pixel data from the background memory addressed through the IPA_BMADDR. Retrieve the pixel data from the background LUT if the background pixel format is indirect.
4) Extend the background pixel value to a 32-bit value, and calculate the alpha channel value according to the BAVCA bits in the IPA_BPCTL register.
5) Blend the processed foreground and background pixel data.
6) Compress the pixel data into the value with the destination pixel format determined by the DPF bits in the IPA_DPCTL register.
7) Write the converted pixel data into the destination memory addressed through the IPA_DMADDR.

Three eight-doubleword depth 64-bit FIFOs are implemented for the foreground, background and four-doubleword depth 64-bit FIFO is for the destination pixel data processing. The foreground and background FIFO are buffers to store the data reading from the corresponding source memory and the destination FIFO is pushed with the processed pixel data which is ready to write into the destination memory when the AXI bus is idle.

If the PFCM bits in the IPA_CTL register is configured to ‘00’ or ‘01’ to copy or convert foreground image to the destination image, only the foreground FIFO and destination FIFO are activated. If the IPA operates to fill up the destination image with the specific color, none of these three FIFOs is activated.

### 35.5.2. Foreground and background LUT

Two LUTs are implemented in the IPA to store the pixel value for the usage of the indirect pixel format. The pixel value must be written into the LUT before the IPA transfer is enabled

when the pixel format is indirect. The pixel value in the LUT can be updated in two ways:

- Automatically loading:

  Enable the FLLEN/BLLEN bit in the IPA_FPCTL/IPA_BPCTL register. The FCNP or BCNP bits in the IPA_FPCTL or IPA_BPCTL register define the number of pixels to be loaded, which is equal to FCNP+ 1 or BCNP + 1.

- Software program:

  The pixel data is written into the corresponding memory address through the IPA AHB slave interface. The base address offset of foreground LUT is 0x0400, and the base address offset of background LUT is 0x0800.

Two pixel formats are supported for the LUTs, including 'ARGB8888' and 'RGB888', which is determined by the FLPF or BLPF bit in the IPA_FPCTL or IPA_BPCTL register, as listed in the ***Table 35-3. Foreground and background CLUT pixel format***.

**Table 35-3. Foreground and background CLUT pixel format**

| BLPF/FLPF | LUT pixel format | Memory address | | | |
|---|---|---|---|---|---|
| | | base + 0x3 | base + 0x2 | base + 0x1 | base + 0x0 |
| 0 | ARGB8888 | $A_0$[7:0] | $R_0$[7:0] | $G_0$[7:0] | $B_0$[7:0] |
| 1 | RGB888 | $R_3$[7:0] | $G_3$[7:0] | $B_3$[7:0] | $R_2$[7:0] |
| | | $G_2$[7:0] | $B_2$[7:0] | $R_1$[7:0] | $G_1$[7:0] |
| | | $B_1$[7:0] | $R_0$[7:0] | $G_0$[7:0] | $B_0$[7:0] |

**Note:** If the pixel format is 'RGB888', the alpha value is fixed to 0xFF when updating the pixel data in the LUT.

### 35.5.3. Foreground and background pixel channel extension (PCE)

In the IPA pixel-format-convert mode with pixel conversion, the foreground (and background) pixel values are extended from the foreground or background pixel format to the 'ARGB8888' format.

The FPF and BPF bits in the IPA_FPCTL and IPA_BPCTL register determine the pixel format of the foreground and background image, as listed in the ***Table 35-4. Foreground and background pixel format***.

A pixel consists of five channels:

- Alpha channel: opacity, 0x00: transparent; 0xFF: opaque.
- Red channel: redness, 0x00 No red, 0xFF: fully red.
- Green channel: greenness, 0x00 No green, 0xFF: fully green.
- Blue channel: blueness, 0x00 No blue, 0xFF: fully blue.
- Luminance channel: In the IPA, the value of the luminance channel is used as an index to retrieve the pixel data from the foreground or background LUT.

**Table 35-4. Foreground and background pixel format**

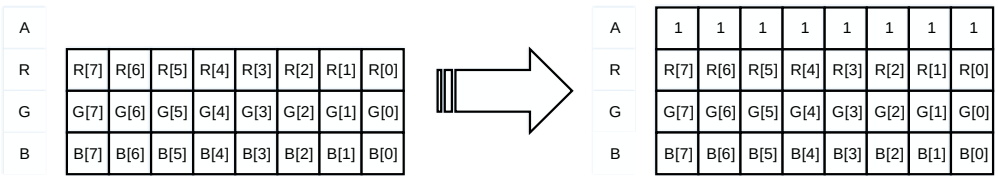
| BPF[3:0]/FPF[3:0] | Pixel format | Memory address | | | |
|---|---|---|---|---|---|
| | | base + 0x3 | base + 0x2 | base + 0x1 | base + 0x0 |
| 0000 | ARGB8888 | $A_0$[7:0] | $R_0$[7:0] | $G_0$[7:0] | $B_0$[7:0] |
| 0001 | RGB888 | $R_3$[7:0] | $G_3$[7:0] | $B_3$[7:0] | $R_2$[7:0] |
| | | $G_2$[7:0] | $B_2$[7:0] | $R_1$[7:0] | $G_1$[7:0] |
| | | $B_1$[7:0] | $R_0$[7:0] | $G_0$[7:0] | $B_0$[7:0] |
| 0010 | RGB565 | $R_1$[4:0]$G_1$[5:3] | $G_1$[2:0]$B_1$[4:0] | $R_0$[4:0]$G_0$[5:3] | $G_0$[2:0]$B_0$[4:0] |
| 0011 | ARGB1555 | $A_1$[0]$R_1$[4:0]$G_1$[4:3] | $G_1$[2:0]$B_1$[4:0] | $A_0$[0]$R_0$[4:0]$G_0$[4:3] | $G_0$[2:0]$B_0$[4:0] |
| 0100 | ARGB4444 | $A_1$[3:0]$R_1$[3:0] | $G_1$[3:0]$B_1$[3:0] | $A_0$[3:0]$R_0$[3:0] | $G_0$[3:0]$B_0$[3:0] |
| 0101 | L8 | $L_3$[7:0] | $L_2$[7:0] | $L_1$[7:0] | $L_0$[7:0] |
| 0110 | AL44 | $A_3$[3:0]$L_3$[3:0] | $A_2$[3:0]$L_2$[3:0] | $A_1$[3:0]$L_1$[3:0] | $A_0$[3:0]$L_0$[3:0] |
| 0111 | AL88 | $A_1$[7:0] | $L_1$[7:0] | $A_0$[7:0] | $L_0$[7:0] |
| 1000 | L4 | $L_7$[3:0]$L_6$[3:0] | $L_5$[3:0]$L_4$[3:0] | $L_3$[3:0]$L_2$[3:0] | $L_1$[3:0]$L_0$[3:0] |
| 1001 | A8 | $A_3$[7:0] | $A_2$[7:0] | $A_1$[7:0] | $A_0$[7:0] |
| 1010 | A4 | $A_7$[3:0]$A_6$[3:0] | $A_5$[3:0]$A_4$[3:0] | $A_3$[3:0]$A_2$[3:0] | $A_1$[3:0]$A_0$[3:0] |
| 1011 | YUV444* | $Y_3$[7:0] | $U_3$[7:0] | $V_3$[7:0] | $Y_2$[7:0] |
| | | $U_2$[7:0] | $V_2$[7:0] | $Y_1$[7:0] | $U_1$[7:0] |
| | | $V_1$[7:0] | $Y_0$[7:0] | $U_0$[7:0] | $V_0$[7:0] |
| 1100 | UYVY422* | $Y_1$[7:0] | $V_{01}$[7:0] | $Y_0$[7:0] | $U_{01}$[7:0] |
| 1101 | VYUY422* | $Y_1$[7:0] | $U_{01}$[7:0] | $Y_0$[7:0] | $V_{01}$[7:0] |
| 1110 | YUV420* | … | … | … | … |
| | | $V_{2367}$[7:0] | $U_{2367}$[7:0] | $V_{0145}$[7:0] | $U_{0145}$[7:0] |
| | | … | … | … | … |
| | | $Y_7$[7:0] | $Y_6$[7:0] | $Y_5$[7:0] | $Y_4$[7:0]@line2 |
| | | $Y_3$[7:0] | $Y_2$[7:0] | $Y_1$[7:0] | $Y_0$[7:0]@line1 |
| 1111 | YVU420* | … | … | … | … |
| | | $U_{2367}$[7:0] | $V_{2367}$[7:0] | $U_{0145}$[7:0] | $V_{0145}$[7:0] |
| | | … | … | … | … |
| | | $Y_7$[7:0] | $Y_6$[7:0] | $Y_5$[7:0] | $Y_4$[7:0]@line2 |
| | | $Y_3$[7:0] | $Y_2$[7:0] | $Y_1$[7:0] | $Y_0$[7:0]@line1 |

**Note*:** YUV format are only supported in foreground image.

If the pixel format is ‘RGB888’, the alpha channel value is equal to 0xFF when extending the pixel data, as shown in ***Figure 35-2. Pixel extension from ‘RGB888’ to ‘ARGB8888’***.

**Figure 35-2. Pixel extension from 'RGB888' to 'ARGB8888'**

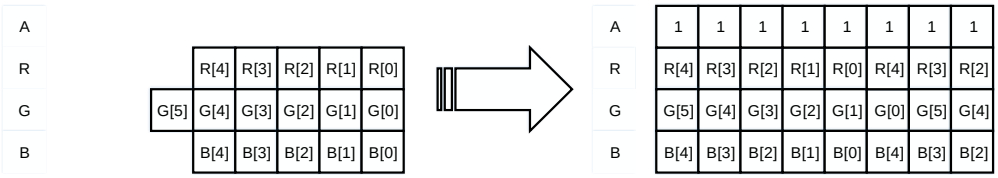
- RGB888 → ARGB8888




If the pixel format is 'RGB565', the alpha channel value is equal to 0xFF when extending the pixel data. The red, green and blue channel value is extended to 8-bit by setting the MSBs to the original bits and copying the MSBs to the LSBs, as shown in ***Figure 35-3. Pixel extension from 'RGB565' to 'ARGB8888'***.

**Figure 35-3. Pixel extension from 'RGB565' to 'ARGB8888'**

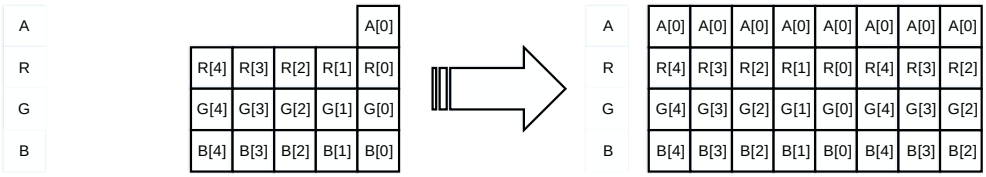
- RGB565 → ARGB8888



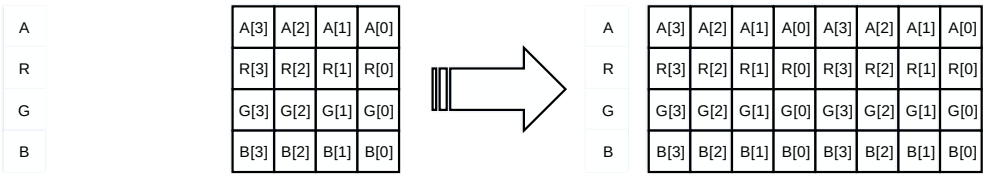

If the pixel format is 'ARGB1555' or 'ARGB4444', The value of every channel is extended to 8-bit by setting the MSBs to the original bits and copying the MSBs to the LSBs, as shown in the ***Figure 35-4. Pixel extension from 'ARGB1555' or 'ARGB4444' to 'ARGB8888'***.

**Figure 35-4. Pixel extension from 'ARGB1555' or 'ARGB4444' to 'ARGB8888'**

- ARGB1555 → ARGB8888




- ARGB4444 → ARGB8888




If the pixel format is 'L8' or 'L4', the pixel data is retrieved from the LUT with the 8-bit luminance channel value (MSBs filled with '0' when 'L4').

If the pixel format is 'AL44', only the red, green and blue channel values are retrieved from the LUT with the 8-bit luminance channel value (MSBs filled with '0'). And the alpha channel value is extended to 8-bit by setting the MSBs to the original bits and copying the MSBs to the LSBs.

If the pixel format is 'AL88', only the red, green and blue channel values are retrieved from

the LUT with the 8-bit luminance channel value.

If the pixel format is 'A8', the red, green and blue channel values are separately equal to the FPDRV or BPDRV bits, FPDGV or BPDGV bits and FPDBV or BPDBV bits in the IPA_FPV or IPA_BPV register.

If the pixel format is 'A4', the alpha channel value is extended to 8-bit by setting the MSBs to the original bits and copying the MSBs to the LSBs. The red, green and blue channel values are separately equal to the FPDRV or BPDRV bits, FPDGV or BPDGV bits and FPDBV or BPDBV bits in the IPA_FPV or IPA_BPV register.

The PCE also support color conversion from YUV/YCbCr to ARGB8888 format. The following equations are used to perform this process. The constants will be stored in the IPA_CSCC_CFGx (x=0…2) control registers as two's compliment values, which allow flexibility in the implementation and differences in the video encode and decode operations. In addition, it also provides a software mechanism to manipulate brightness or contrast.

$$R = C0(Y+Yoffset) + C1(V+UVoffset) \quad (35\text{-}1)$$

$$G = C0(Y+Yoffset) + C3(U+UVoffset) + C2(V+UVoffset) \quad (35\text{-}2)$$

$$B = C0(Y+Yoffset) + C4(U+UVoffset) \quad (35\text{-}3)$$

***Table 35-5. Expected coefficients for YUV and YCbCr modes*** indicates the expected coefficients for YUV and YCbCr modes.

**Table 35-5. Expected coefficients for YUV and YCbCr modes**

| Coff | YUV | YCbCr |
|---|---|---|
| YOFF | 0x000 | 0x1F0 (-16) |
| UVOFF | 0x000 | 0x180 (-128) |
| C0 | 0x100 (1.00) | 0x129 (1.164) |
| C1 | 0x123 (1.140) | 0x198 (1.596) |
| C2 | 0x76B (-0.581) | 0x72F (-0.813) |
| C3 | 0x79B (-0.394) | 0x79B (-0.392) |
| C4 | 0x208 (2.032) | 0x204 (2.017) |

Three algorithms are supported to modulate the alpha channel value, which is determined by the FAVCA or BAVCA bits in the IPA_FPCTL or IPA_BPCTL register, as described in ***Table 35-6. Alpha channel value modulation***.

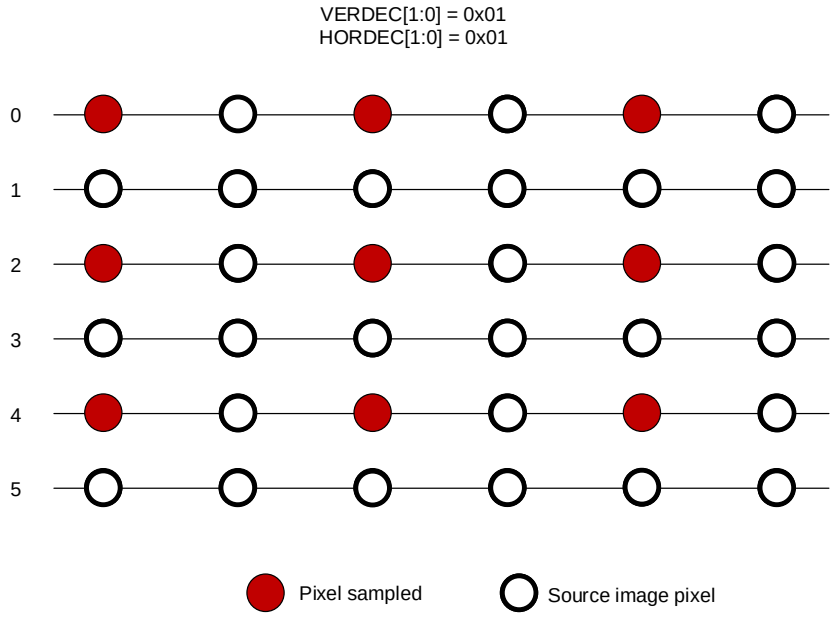
**Table 35-6. Alpha channel value modulation**

| FAVCA[1:0]/BAVCA[1:0] | Alpha calculation algorithm |
|---|---|
| 00/11 | No effect, equal to the original value |
| 01 | Equal to the FPDAV or BPDAV bits in the IPA_FPCTL or IPA_BPCTL register |
| 10 | Equal to the FPDAV or BPDAV bits multiplied by the original alpha value and divided by 255 |

### 35.5.4. Foreground channel scaling

IPA implements a bilinear scaling filter to adjust the input image to different resolutions in foreground channel. The HORDEC and VERDEC in IPA_DPCTL register define the horizontal and vertical pre decimation filter control. The XSCALE and YSCALE in IPA_BSCTL register provide the the X and Y scaling factor for the Foreground. The maximum down scaling factors of decimation filter and bilinear filter are 8 and 2 respectively, so that the maximum scaling factor can be up to 16. The detailed sampling method of the decimation filter is shown in ***Figure 35-5. Sampling method of decimation filter***.

**Figure 35-5. Sampling method of decimation filter**

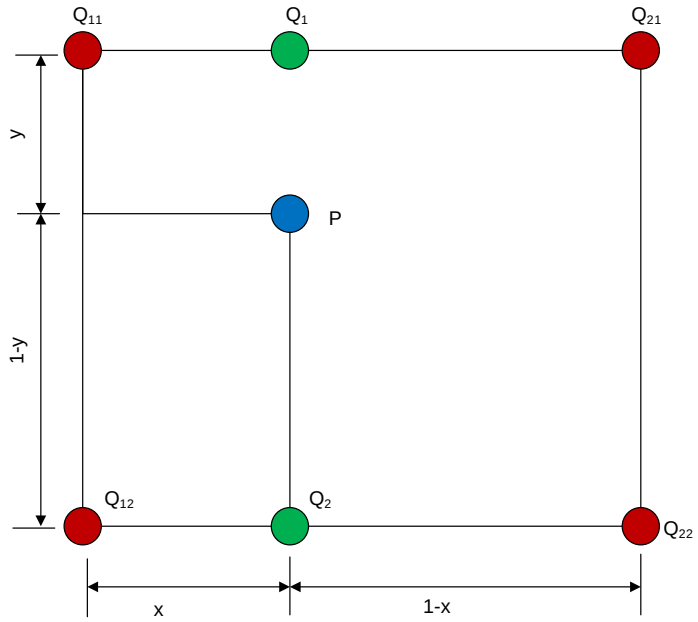



The maximum down scaling factor is 1/2 and the maximum up scaling factor is 2^12 for each axis. In order to realize the scaling function, the reciprocal of the scale factor should be written to IPA_BSCTL register. For example:

```
REG32(IPA_BSCTL) = 0x20002000;      //1/2x scaling (0x2.0)
REG32(IPA_BSCTL) = 0x18001800;      //2/3x scaling (0x1.8)
REG32(IPA_BSCTL) = 0x08000800;      //2x scaling (0x0.8)
REG32(IPA_BSCTL) = 0x04000400;      //4x scaling (0x0.4)
```

The bilinear filter is the weighted average of the four nearest pixels. Suppose P is the output pixel and $Q_{11}$, $Q_{12}$, $Q_{21}$, $Q_{22}$ are the surrounding four source image pixels. As is shown in ***Figure 35-6. Bilinear Scaling diagram***.

**Figure 35-6. Bilinear Scaling diagram**




The value of output pixel P is computed by the equation below.

$$P = Q_1(1-y) + Q_2*y \quad (35\text{-}4)$$

Where the equations for $Q_1$ and $Q_2$ are as follows:

$$Q_1 = Q_{11}*(1-x) + Q_{21}*x \quad (35\text{-}5)$$

$$Q_2 = Q_{12}*(1-x) + Q_{22}*x \quad (35\text{-}6)$$

The Scaling are only effecitive when PFCM eques to 2'b01 or 2'b10.

### 35.5.5. Blending

When the IPA operates to convert and blend the foreground and background images to the destination image, the foreground and background pixel data after extending are blended by pair to get a 32-bit pixel value.

The alpha channel value is blended on the base of the following equations ($A_F$ is the foreground alpha value, $A_B$ is the background alpha value):

$$A_{mix} = \frac{A_F \times A_B}{255} \quad (35\text{-}7)$$

$$A_{blend} = A_F + A_B - A_{mix} \quad (35\text{-}8)$$

The red, green and blue channel value are blended on the base of the following equations ($R_F$, $G_F$, $B_F$ is the foreground red, green and blue value; $R_B$, $G_B$, $B_B$ is the background red, green and blue value):

$$R_{blend} = \frac{R_F \times A_F + R_B \times A_B - R_B \times A_{mix}}{A_{blend}} \quad (35\text{-}9)$$

$$G_{blend} = \frac{G_F \times A_F + G_B \times A_B - G_B \times A_{mix}}{A_{blend}} \quad (35\text{-}10)$$

$$B_{blend} = \frac{B_F \times A_F + B_B \times A_B - B_B \times A_{mix}}{A_{blend}} \quad (35\text{-}11)$$

**Note:** 1) The quotient of the division is rounded down to the nearest integer.

2) If the $A_{blend}$ is equal to zero, the $R_{blend}$, $G_{blend}$ and $B_{blend}$ is equal to '0xFF'.

3) The width and height of the background and the destination shall be consistent.

### 35.5.6. Destination pixel channel compression (PCC)

In the IPA pixel-format-convert mode with pixel conversion, the pixel data need to be compressed from the 'ARGB8888' format into the destination pixel format before they are written into the destination memory.

The DPF bits in the IPA_DPCTL register determine the pixel format of the destination image, as listed in ***Table 35-7. Destination pixel format***.

**Table 35-7. Destination pixel format**

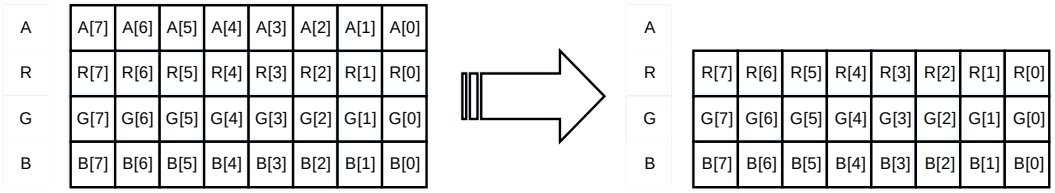
| DPF[2:0] | Pixel format | Memory address | | | |
|---|---|---|---|---|---|
| | | base + 0x3 | base + 0x2 | base + 0x1 | base + 0x0 |
| 000 | ARGB8888 | $A_0$[7:0] | $R_0$[7:0] | $G_0$[7:0] | $B_0$[7:0] |
| 001 | RGB888 | $R_3$[7:0] | $G_3$[7:0] | $B_3$[7:0] | $R_2$[7:0] |
| | | $G_2$[7:0] | $B_2$[7:0] | $R_1$[7:0] | $G_1$[7:0] |
| | | $B_1$[7:0] | $R_0$[7:0] | $G_0$[7:0] | $B_0$[7:0] |
| 010 | RGB565 | $R_1$[4:0]$G_1$[5:3] | $G_1$[2:0]$B_1$[4:0] | $R_0$[4:0]$G_0$[5:3] | $G_0$[2:0]$B_0$[4:0] |
| 011 | ARGB1555 | $A_1$[0]$R_1$[4:0]$G_1$[4:3] | $G_1$[2:0]$B_1$[4:0] | $A_0$[0]$R_0$[4:0]$G_0$[4:3] | $G_0$[2:0]$B_0$[4:0] |
| 100 | ARGB4444 | $A_1$[3:0]$R_1$[3:0] | $G_1$[3:0]$B_1$[3:0] | $A_0$[3:0]$R_0$[3:0] | $G_0$[3:0]$B_0$[3:0] |

**Note:** If the PFCM bits in the IPA_CTL register are equal to '00' (copy the foreground image to the destination image), the DPF bits have no meaning, and the FPF bits in the IPA_FPCTL register determine the bit number per pixel for both the source and destination.

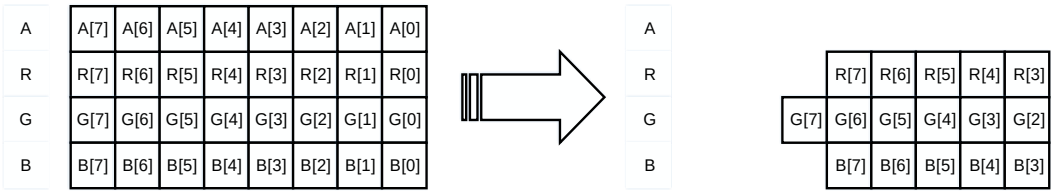
As shown in ***Figure 35-7. Pixel compression***, the destination compression is performed by discarding the LSBs.

**Figure 35-7. Pixel compression**

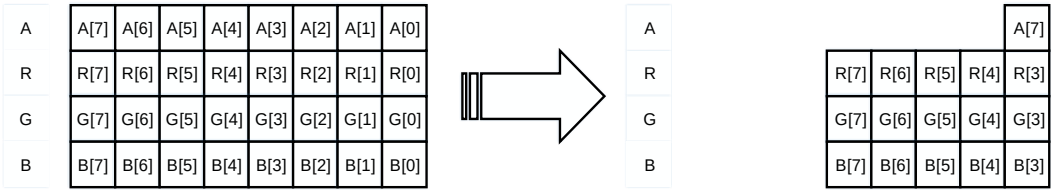
- ARGB8888 → RGB888



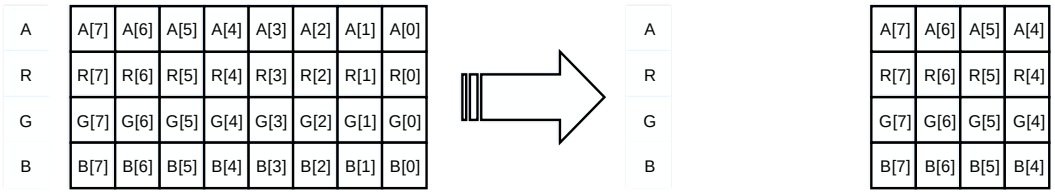

- ARGB8888 → RGB565




- ARGB8888 → ARGB1555




- ARGB8888 → ARGB4444




## 35.5.7. Rotation

The IPA support rotations of 0, 90, 180, and 270 degrees. when ROT bits in IPA_DPCTL are configured, the IPA read input image line by line and put each converted output pixel into the generated output pixel address after rotation.

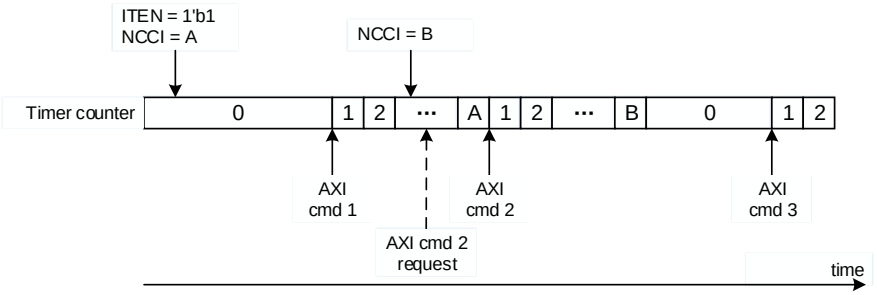
## 35.5.8. Inter-timer

To reduce the AXI bandwidth usage of IPA AXI master interface, a timer is implemented to insert a number of clocks between two consecutive AXI commands during IPA transmission and LUT automatic loading.

The internal timer is enabled by setting the ITEN bit in the IPA_ITCTL register. The NCCI bits in the IPA_ITCTL register define the minimum number of clock inserted between two consecutive AXI write or read address channel commands, and these bits have no meaning when the timer is disabled.

Updating the NCCI bits when the ITEN bit is enabled have no effect until the current counting is completed, as shown in ***Figure 35-8. Inter timer operation***.

**Figure 35-8. Inter timer operation**




## 35.5.9. Line mark

The marked line number can be set by configuring the LM bits in the IPA_LM register. As soon as the last pixel data of the line mark has been written into the destination memory, the TLMIF bit in the IPA_INTF register is asserted to detailing the progression of the IPA transfer.

**Note:** If the LM bits are equal to zero, no line mark flag is asserted during the transmission.

## 35.5.10. Transfer flow

The foreground/background LUT automatically loading is enabled by setting the FLLEN/BLLEN bit in the IPA_FPCTL/IPA_BPCTL register. Once the loading transfer is launched, the FLLEN/BLLEN bit is used as a transmission flag and writing ‘0’ to the FLLEN/BLLEN bit has no meaning. The FLLEN/BLLEN can be automatically cleared when the loading is finished.

The IPA transfer is enabled by setting the TEN bit in the IPA_CTL register. Once the IPA transfer is launched, the TEN bit is used as a transmission flag and writing ‘0’ to it has no meaning. The TEN bit can be automatically cleared when the IPA transfer is finished.

At any time, the foreground/background LUT automatic loading and IPA transfer can be hanged up by setting the THU bit in the IPA_CTL register. The LUT loading and IPA transfer is paused until the THU bit is cleared by software. When none of the foreground/background LUT automatically loading and IPA transfer is enabled, setting the THU bit has no effect and the THU bit is read as 0.

The foreground/background LUT automatic loading and IPA transfer can be stopped by setting the TST bit in the IPA_CTL register. The LUT loading or IPA transfer is stopped immediately by resetting the FLLEN/BLLEN bit in the IPA_FPCTL/IPA_BPCTL register or the TEN bit in the IPA_CTL register even though the LUT loading or IPA transfer is being hanged up. The TST bit is automatically reset when the current transfer is disabled. When none of the foreground/background LUT automatic loading and IPA transfer is enabled, setting the TST bit has no effect and the TST bit is read as 0.

Only one of the foreground LUT loading, background LUT loading and IPA transfer can be working at a time. For example, when the IPA transfer is ongoing, setting the FLLEN or BLLEN bit has no effect and the FLLEN and BLLEN bit is automatically reset.

### 35.5.11. Configuration

Before launching any transfers, it is necessary to read the TEN, FLLEN and BLLEN bit to check whether the IPA transfer or the LUT loading is active. If one of them is ongoing, set the TST bit to stop it or wait it finished. When all of the TEN, FLLEN and BLLEN bit are read as 0, starting a new transfer is allowed.

#### Foreground LUT loading

When starting a new foreground LUT loading, it is recommended to respect the following steps:

1. Configure the IPA_FLMADDR register to set the foreground LUT memory base address.
2. Configure the FLPF bit in the IPA_FPCTL register to set the foreground LUT pixel format.
3. Configure the FCNP bits in the IPA_FPCTL register to set the number of pixel in the foreground LUT to be loaded.
4. Configure the needed enable bit for wrong configuration interrupt, LUT loading finish interrupt, LUT access conflict interrupt and transfer access error interrupt in the IPA_CTL register.
5. Configure the FLLEN bit with ‘1’ in the IPA_FPCTL register to enable the foreground LUT automatically loading.

#### Background LUT loading

When starting a new background LUT loading, it is recommended to respect the following steps:

1. Configure the IPA_BLMADDR register to set the background LUT memory base address.
2. Configure the BLPF bit in the IPA_BPCTL register to set the background LUT pixel format.
3. Configure the BCNP bits in the IPA_BPCTL register to set the number of pixel in the background LUT to be loaded.
4. Configure the needed enable bit for wrong configuration interrupt, LUT loading finish interrupt, LUT access conflict interrupt and transfer access error interrupt in the IPA_CTL register.
5. Configure the BLLEN bit with ‘1’ in the IPA_BPCTL register to enable the background LUT automatically loading.

#### IPA transfer

When starting a new IPA transfer, the configuration steps corresponding with the pixel format convert mode are as follows:

**Copy the foreground image to the destination image**

1. Configure the IPA_FMADDR and IPA_DMADDR register to set the foreground and destination memory base address.
2. Configure the FPF bits in the IPA_FPCTL register to set the foreground pixel format.
3. Configure the FLOFF and DLOFF bits in the IPA_FLOFF and IPA_DLOFF register to set the foreground and destination line offset.
4. Configure the LM bits in the IPA_LM register to set the line mark if needed.
5. Configure the WIDTH and HEIGHT bits in the IPA_IMS register to set the image size.
6. Configure the needed enable bit for wrong configuration interrupt, LUT access conflict interrupt, transfer line mark interrupt, full transfer finish interrupt and transfer access error interrupt in the IPA_CTL register.
7. Configure the TEN bit with '1' in the IPA_CTL register to enable the IPA transfer.

### Convert foreground image to the destination image

If the foreground pixel format is indirect, the pixel data must be loaded into the foreground LUT before starting the IPA transfer. The LUT automatic loading procedure is described in the ***Foreground LUT loading***.

1. Configure the IPA_FMADDR and IPA_DMADDR register to set the foreground and destination memory base address.
2. Configure the FAVCA and FPF bits in the IPA_FPCTL register to set the foreground alpha value calculation algorithm and the foreground pixel format. Configure the IPA_CSCC_CFGx (x=0…2) when foreground pixel format is YUV/YCbCr.
3. Configure the pre-defined pixel value, including alpha, red, green and blue value in the IPA_FPCTL and IPA_FPV register if the foreground format is not ARGBxxxx type.
4. Configure the DPF bits in the IPA_DPCTL register to set the destination pixel format.
5. Configure the FLOFF and DLOFF bits in the IPA_FLOFF and IPA_DLOFF register to set the foreground and destination line offset.
6. Configure the VERDEC/HORDEC bits in the IPA_DPCTL, XSCAL/YSCAL bits in IPA_BSCTL and DWIDTH/DHEIGHT bits in the IPA_DIMS if scaling is needed.
7. Configure the ROT bits in the IPA_DPCTL when rotation is needed.
8. Configure the LM bits in the IPA_LM register to set the line mark if needed.
9. Configure the WIDTH and HEIGHT bits in the IPA_IMS register to set the image size.
10. Configure the needed enable bit for wrong configuration interrupt, LUT access conflict interrupt, transfer line mark interrupt, full transfer finish interrupt and transfer access error interrupt in the IPA_CTL register.
11. Configure the TEN bit with '1' in the IPA_CTL register to enable the IPA transfer.

### Convert and blend the foreground and background images to the destination image

If the foreground or background pixel format is indirect, the pixel data must be loaded into the corresponding LUT before starting the IPA transfer. The foreground and background LUT automatically loading procedure is described in the ***Foreground LUT loading*** and ***Background LUT loading***.

1. Configure the IPA_FMADDR, IPA_BMADDR and IPA_DMADDR register to set the

foreground, background and destination memory base address.

2. Configure the FAVCA and FPF bits in the IPA_FPCTL register to set the foreground alpha value calculation algorithm and the foreground pixel format. Configure the IPA_CSCC_CFGx (x=0…2) when foreground pixel format is YUV/YCbCr.
3. Configure the foreground pre-defined pixel value, including alpha, red, green and blue value in the IPA_FPCTL and IPA_FPV register if the foreground format is not ARGBxxxx type.
4. Configure the BAVCA and BPF bits in the IPA_BPCTL register to set the background alpha value calculation algorithm and the background pixel format. Configure the IPA_CSCC_CFGx (x=0…2) when foreground pixel format is YUV/YCbCr.
5. Configure the background pre-defined pixel value, including alpha, red, green and blue value in the IPA_BPCTL and IPA_BPV register if the background format is not ARGBxxxx type.
6. Configure the DPF bits in the IPA_DPCTL register to set the destination pixel format.
7. Configure the FLOFF, BLOFF and DLOFF bits in the IPA_FLOFF, IPA_BLOFF and IPA_DLOFF register to set the foreground, background and destination line offset.
8. Configure the LM bits in the IPA_LM register to set the line mark if needed.
9. Configure the WIDTH and HEIGHT bits in the IPA_IMS register to set the image size.
10. Configure the VERDEC/HORDEC bits in the IPA_DPCTL, XSCAL/YSCAL bits in IPA_BSCTL and DWIDTH/DHEIGHT bits in the IPA_DIMS if scaling is needed.
11. Configure the ROT bits in the IPA_DPCTL when rotation is needed.
12. Configure the needed enable bit for wrong configuration interrupt, LUT access conflict interrupt, transfer line mark interrupt, full transfer finish interrupt and transfer access error interrupt in the IPA_CTL register.
13. Configure the TEN bit with ‘1’ in the IPA_CTL register to enable the IPA transfer.

### Fill up the destination image with a specific color

1. Configure the IPA_DMADDR register to set the destination memory base address.
2. Configure the DPF bits in the IPA_DPCTL register to set the destination pixel format.
3. Configure the destination pre-defined pixel value, including alpha, red, green and blue value in the IPA_DPV register.
4. Configure the DLOFF bits in the IPA_DLOFF register to set the destination line offset.
5. Configure the LM bits in the IPA_LM register to set the line mark if needed.
6. Configure the WIDTH and HEIGHT bits in the IPA_IMS register to set the image size.
7. Configure the needed enable bit for wrong configuration interrupt, LUT access conflict interrupt, transfer line mark interrupt, full transfer finish interrupt and transfer access error interrupt in the IPA_CTL register.
8. Configure the TEN bit with ‘1’ in the IPA_CTL register to enable the IPA transfer.

### Configuration rules

The IPA configuration must respect a number of rules, otherwise the transfer or loading is automatically reset and the WCFIF bit in the IPA_INTF register is asserted immediately after it is enabled. The rules are described as follows:

When the foreground LUT automatically loading is enabled:

- The FLMADDR bits in the IPA_FLMADDR register must be 32-bit alignment when the FLPF bit in the IPA_FPCTL register is equal to ‘0’.

When the background LUT automatically loading is enabled:

- The BLMADDR bits in the IPA_BLMADDR register must be 32-bit alignment when the BLPF bit in the IPA_BPCTL register is equal to ‘0’.

When the IPA transfer is enabled:

1) The FMADDR bits in the IPA_FMADDR register must be 32-bit alignment when the FPF bits in the IPA_FPCTL register are ‘ARGB8888’, ’UYVY422’, ‘VYUY422’, ’YUV420’ or ‘YVU420’ and be 16-bit alignment when the FPF bits are ‘RGB565’, ‘ARGB1555’, ‘ARGB4444’ or ‘AL88’ .
2) The FLOFF bits in the IPA_FLOFF register must be even when the FPF bits in the IPA_FPCTL register are ‘A4’, ‘L4’, ‘UYVY422’ or ‘VYUY422’, and be 4-bit alignment when the FPF bits in the IPA_FPCTL register are ’YUV420’ or ‘YVU420’.
3) The BMADDR bits in the IPA_BMADDR register must be 32-bit alignment when the BPF bits in the IPA_BPCTL register are ‘ARGB8888’, and be 16-bit alignment when the BPF bits are ‘RGB565’, ‘ARGB1555’, ‘ARGB4444’ or ‘AL88’ .
4) The BLOFF bits in the IPA_BLOFF register must be even when the BPF bits in the IPA_BPCTL register are ‘A4’ or ‘L4’.
5) The FPF bits in the IPA_FPCTL register must be valid and less than or equal to ‘0b1111’ when PFCM bits are not ‘0b00’ or ‘0b11’, and be valid and less than or equal to ‘0b1101’ when PFCM bits are ‘0b00’.
6) The BPF bits in the IPA_BPCTL register must be valid and less than or equal to ‘0b1010’.
7) The DPF bits in the IPA_DPCTL register must be valid and less than or equal to ‘0b100’.
8) The DMADDR bits in the IPA_DMADDR register must be 32-bit alignment when the DPF bits in the IPA_DPCTL register are ‘ARGB8888’ and be 16-bit alignment when the DPF bits are ‘RGB565’, ‘ARGB1555’, ‘ARGB4444’.
9) The DLOFF bits in the IPA_DLOFF register must be even when the FPF bits in the IPA_FPCTL register are ‘A4’ or ‘L4’.
10) The WIDTH bits in the IPA_IMS register must be even when the FPF bits in the IPA_FPCTL register are ‘A4’, ‘L4’, ‘UYVY422’ or ‘VYUY422’, and be 4-bit-alignment when the FPF bits in the IPA_FPCTL register are ’YUV420’ or ‘YVU420’.
11) The WIDTH bits in the IPA_IMS register must be even when the BPF bits in the IPA_BPCTL register are ‘A4’ or ‘L4’.
12) The WIDTH bits in the IPA_IMS register must be greater than zero.
13) The HEIGHT bits in the IPA_IMS register must be greater than zero.
14) The XSCALE and YSCALE bits in the IPA_BSCTL register must be greater than zero and less than ‘0x2001’.
15) When the FIIMEN bit is set, the FPF bits in the IPA_FPCTL register must not be ‘YUV420’ and ‘YVU420’, and the EFUVMADDR bits in the IPA_EF_UV_MADDR register must be 32-bit alignment when the FPF bits in the IPA_FPCTL register are ‘ARGB8888’, ’UYVY422’, ‘VYUY422’, and be 16-bit alignment when the FPF bits are ‘RGB565’, ‘ARGB1555’, ‘ARGB4444’ or ‘AL88’.

16) The EFUVMADDR bits in the IPA_EF_UV_MADDR register must be 16-bit alignment when the FPF bits in the IPA_FPCTL register are ‘YUV420’ or ‘YVU420’.
17) When the forechannel scaling is enabled, that is, the HORDEC, VERDEC, XSCALE, YSCALE are not default value. The DWIDTH bits in the IPA_DIMS must be even when the BPF bits in the IPA_BPCTL register are ‘A4’, ‘L4’.
18) When the forechannel scaling and rotation function are enabled, that is, the HORDEC , VERDEC, XSCALE, YSCALE, ROT are not default value. The DHEIGHT bits in the IPA_DIMS must be even when the BPF bits in the IPA_BPCTL register are ‘A4’ , ‘L4’.

When the PFCM bits are equal to ‘00’, only 1), 2), 5), 9), 10), 12), 13) are considerable.
When the PFCM bits are equal to ‘01’, only 1), 2), 5), 7), 8), 10), 12), 13) ,14), 15), 16)are considerable.
When the PFCM bits are equal to ‘10’, all the configuration rules except 9) are considerable.
When the PFCM bits are equal to ‘11’, only 7), 8), 12), 13) are considerable.

## 35.6. Interrupts

There are six interrupt events connected to the IPA interrupt, including wrong configuration interrupt, LUT loading finish interrupt, LUT access conflict interrupt, transfer line mark interrupt, full transfer finish interrupt and transfer access error interrupt. An IPA interrupt can be produced when any interrupt events occurs.

Each interrupt event has a dedicated flag bit in the IPA_INTF register, a dedicated clear bit in the IPA_INTC register, and a dedicated enable bit in the IPA_CTL register. The relationship is described in the ***Table 35-8. IPA interrupt events***.

**Table 35-8. IPA interrupt events**

| Interrupt event | Flag bit | Enable bit | Clear bit |
|---|---|---|---|
| | IPA_INTF | IPA_CTL | IPA_INTC |
| wrong configuration interrupt | WCFIF | WCFIE | WCFIFC |
| LUT loading finish interrupt | LLFIF | LLFIE | LLFIFC |
| LUT access conflict interrupt | LACIF | LACIE | LACIFC |
| transfer line mark interrupt | TLMIF | TLMIE | TLMIFC |
| full transfer finish interrupt | FTFIF | FTFIE | FTFIFC |
| transfer access error interrupt | TAEIF | TAEIE | TAEIFC |

### Wrong configuration interrupt

The wrong configuration interrupt flag is asserted immediately after the LUT loading or IPA transfer is enabled, when any of the configuration rules listed in the ***Configuration rules*** is broken. The LUT loading or IPA transfer is automatically disabled without launching any access.

When the wrong configuration interrupt flag is asserted and the enabled bit for wrong configuration interrupt is set, an IPA interrupt is generated.

### LUT loading finish interrupt

The LUT loading finish interrupt flag is asserted immediately after the last pixel data has been loaded into the foreground or background LUT. A stop operation during the loading cannot assert the LUT loading finish interrupt flag.

When the LUT loading finish interrupt flag is asserted and the enabled bit for LUT loading finish interrupt is set, an IPA interrupt is generated.

### LUT access conflict interrupt

A number of rules must be respected when accessing the foreground and background LUT by software:

- During the foreground LUT automatic loading, the foreground LUT is forbidden to be accessed by software.
- During the background LUT automatic loading, the background LUT is forbidden to be accessed by software.
- During the IPA transfer with the PFCM bits equal to '0b01' or '0b10', if the foreground pixel format is indirect, the foreground LUT is forbidden to be accessed by software.
- During the IPA transfer with the PFCM bits equal to '0b10', if the background pixel format is indirect, the background LUT is forbidden to be accessed by software.

When one of the above rules is broken, the LUT access conflict interrupt flag is asserted and the software access has no effect (writing access is not be executed, reading access is returned with an invalid value).

When the LUT access conflict interrupt flag is asserted and the enabled bit for the LUT access conflict interrupt is set, an IPA interrupt is generated.

### Transfer line mark interrupt

The transfer line mark interrupt flag is asserted immediately after the last pixel data of the line mark is written into the destination memory. If the LM bits in the IPA_LM register are equal to 0, the transfer line mark interrupt flag will never be asserted during the IPA transmission.

When the transfer line mark interrupt flag is asserted and the enabled bit for the transfer line mark interrupt is set, an IPA interrupt is generated.

### Full transfer finish interrupt

The full transfer finish interrupt flag is asserted immediately after the last pixel data has been written into the destination memory. A stop operation during the IPA transmission cannot assert the full transfer finish interrupt flag.

When the full transfer finish interrupt flag is asserted and the enabled bit for the full transfer finish interrupt is set, an IPA interrupt is generated.

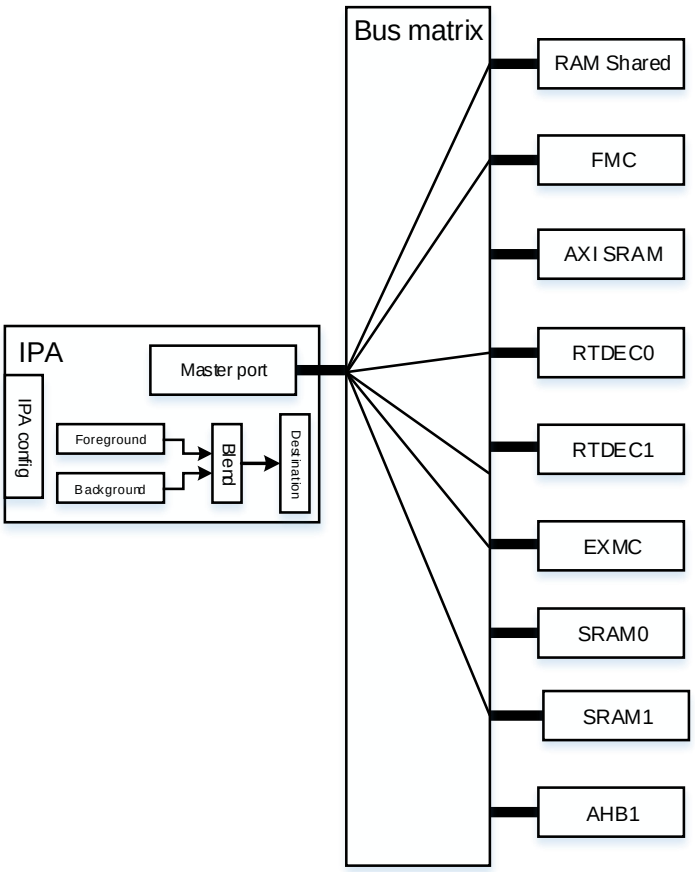
### Transfer access error interrupt

When the address accessed by the IPA is beyond the allowed area, a response error will be received and the transfer (LUT loading or IPA transfer) is disabled immediately without asserting the LUT loading finish interrupt flag or the full transfer finish interrupt flag. The allowed and forbidden access region for IPA is shown in ***Figure 35-9. System connection***

*of IPA*.

When the transfer access error interrupt flag is asserted and the enabled bit for the transfer access error interrupt is set, an IPA interrupt is generated.

**Figure 35-9. System connection of IPA**



## 35.7. Register definition

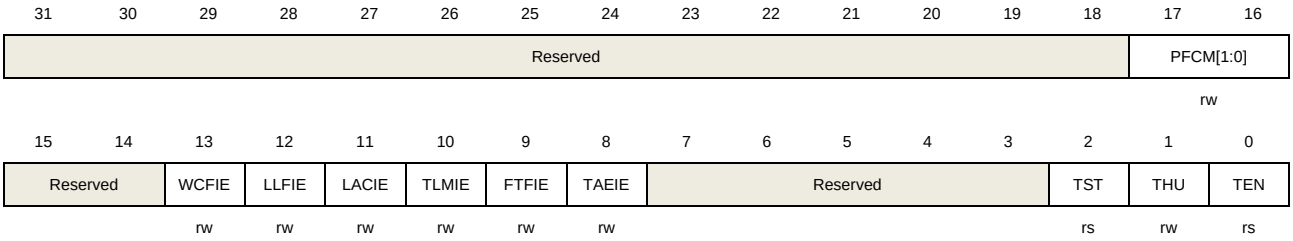
IPA base address: 0x5200 1000

### 35.7.1. Control register (IPA_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17:16 | PFCM[1:0] | Pixel format convert mode<br>Software set and clear.<br>00: Foreground memory to destination memory without pixel format convert<br>01: Foreground memory to destination memory with pixel format convert<br>10: Blending foreground and background memory to destination memory<br>11: Fill up destination memory with specific color<br>These bits can NOT be written when TEN is ‘1’. |
| 15:14 | Reserved | Must be kept at reset value |
| 13 | WCFIE | Enable bit for wrong configuration interrupt<br>Software set and clear<br>0: Disable configuration error interrupt<br>1: Enable configuration error interrupt |
| 12 | LLFIE | Enable bit for LUT loading finish interrupt<br>Software set and clear<br>0: Disable LUT loading finish interrupt<br>1: Enable LUT loading finish interrupt |
| 11 | LACIE | Enable bit for LUT access conflict interrupt<br>Software set and clear<br>0: Disable LUT access conflict interrupt<br>1: Enable LUT access conflict interrupt |
| 10 | TLMIE | Enable bit for transfer line mark interrupt<br>Software set and clear |

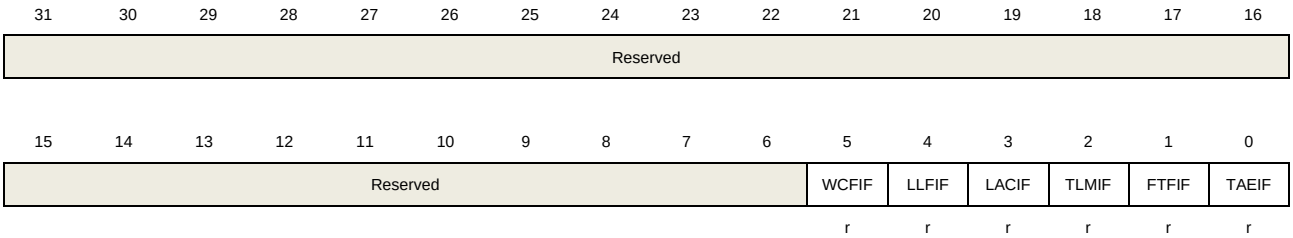
| | | |
|---|---|---|
| | | 0: Disable transfer line mark interrupt<br>1: Enable transfer line mark interrupt |
| 9 | FTFIE | Enable bit for full transfer finish interrupt<br>Software set and clear<br>0: Disable full transfer finish interrupt<br>1: Enable full transfer finish interrupt |
| 8 | TAEIE | Enable bit for transfer access error interrupt<br>Software set and clear<br>0: Disable transfer access error interrupt<br>1: Enable transfer access error interrupt |
| 7:3 | Reserved | Must be kept at reset value |
| 2 | TST | Transfer stop<br>Software set, software and hardware clear.<br>0: No effect<br>1: Stop the current transfer<br>When this bit is enabled, the current transfer (including LUT automatic loading and IPA transfer) is stopped. This bit can be cleared by hardware immediately when the current transfer is disabled. |
| 1 | THU | Transfer hang up<br>Software set, software and hardware clear.<br>0: No effect<br>1: Hang up the current transfer<br>When this bit is enabled, the current transfer (including LUT automatic loading and IPA transfer) is hanged up. When this bit is cleared, the current transfer continues. This bit can be cleared by hardware immediately when the current transfer is disabled. |
| 0 | TEN | Transfer enable<br>Software set, hardware clear.<br>0: Transfer disable<br>1: Transfer enable<br>When this bit is enabled, the IPA transfer is started. This bit is automatically cleared when one of the following situations occurs:<br>- When the TST bit is enabled to stop the current transfer.<br>- When the transfer is fully finished.<br>- When a wrong configuration or a transfer access error is detected.<br>- When the foreground LUT or background LUT is being loaded (FLLEN bit in the IPA_FPCTL register or BLLEN bit in the IPA_BPCTL register is '1'). |

### 35.7.2. Interrupt flag register (IPA_INTF)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5 | WCFIF | Wrong configuration interrupt flag<br>Hardware set, Software cleared by enable ‘WCFIFC’ bit in the IPA_INTC register.<br>0: No wrong configuration is detected when IPA transfer or LUT loading is enable.<br>1: A wrong configuration is detected when IPA transfer or LUT loading is enable. |
| 4 | LLFIF | LUT loading finish interrupt flag<br>Hardware set, software cleared by enable ‘LLFIFC’ bit in the IPA_INTC register.<br>0: No LUT loading finish is detected<br>1: A LUT loading finish is detected |
| 3 | LACIF | LUT access conflict interrupt flag<br>Hardware set, software cleared by enable ‘LACIFC’ bit in the IPA_INTC register.<br>0: No LUT access conflict is detected.<br>1: A LUT access conflict is detected. |
| 2 | TLMIF | Transfer line mark interrupt flag<br>Hardware set, software cleared by enable ‘CTCLIF’ bit in the IPA_INTC register.<br>0: The number of pixel transferred has not exactly reached the line mark<br>1: The number of pixel transferred has exactly reached the line mark |
| 1 | FTFIF | Full transfer finish interrupt flag<br>Hardware set, software cleared by enable ‘CTFIF’ bit in the IPA_INTC register.<br>0: No full transfer finish is detected.<br>1: A full transfer finish is detected. |
| 0 | TAEIF | Transfer access error interrupt flag<br>Hardware set, software cleared by enable ‘CTEIF’ bit in the IPA_INTC register.<br>0: No transfer access error is detected.<br>1: A transfer access error is detected. |

### 35.7.3. Interrupt flag clear register (IPA_INTC)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | WCFIFC | LLFIFC | LACIFC | TLMIFC | FTFIFC | TAEIFC |
| | | | | | | | | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5 | WCFIFC | Clear bit for wrong configuration interrupt flag<br>Software set, hardware clear<br>0: No effect<br>1: Clear wrong configuration interrupt flag |
| 4 | LLFIFC | Clear bit for LUT loading finish interrupt flag<br>Software set, hardware clear<br>0: No effect<br>1: Clear LUT loading finish interrupt flag |
| 3 | LACIFC | Clear bit for LUT access conflict interrupt flag<br>Software set, hardware clear<br>0: No effect<br>1: Clear LUT access conflict interrupt flag |
| 2 | TLMIFC | Clear bit for transfer line mark interrupt flag<br>Software set, hardware clear<br>0: No effect<br>1: Clear transfer line mark interrupt flag |
| 1 | FTFIFC | Clear bit for full transfer finish interrupt flag<br>Software set, hardware clear<br>0: No effect<br>1: Clear full transfer finish interrupt flag |
| 0 | TAEIFC | Clear bit for transfer access error interrupt flag<br>Software set, hardware clear<br>0: No effect<br>1: Clear transfer access error interrupt flag |

### 35.7.4. Foreground memory base address register (IPA_FMADDR)

Address offset: 0x0C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FMADDR[31:0] | Foreground memory base address<br>These bits must be aligned to 8-bit, 16-bit or 32-bit corresponding with the foreground pixel format. If foreground pixel format is ARGB8888, UYVY422, VYUY422 ,YUV420 or YVU420, these bits must be 32-bit aligned; If the foreground pixel format is RGB565, ARGB1555, ARGB4444 or AL88, these bits must be 16-bit aligned. If the above alignment rules are broken, a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

## 35.7.5. Foreground line offset register (IPA_FLOFF)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FLOFF[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:0 | FLOFF[13:0] | Foreground line offset<br>These bits indicate the number of pixel between the last pixel of the current line and the first pixel of the next line. If the foreground pixel format is A4 or L4, the FLOFF must be configured to be an even number, and if the foreground pixel format is YUV420 or YVU420, the FLOFF must be configured 4-bit alignment otherwise a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

## 35.7.6. Background memory base address register (IPA_BMADDR)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BMADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BMADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | BMADDR[31:0] | Background memory base address<br>These bits must be aligned to 8-bit, 16-bit or 32-bit corresponding with the background pixel format. If background pixel format is ARGB8888, UYVY422 and VYUY422, these bits must be 32-bit aligned; If the background pixel format is RGB565, ARGB1555, ARGB4444 or AL88, these bits must be 16-bit aligned. If the above alignment rules are broken, a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

## 35.7.7. Background line offset register (IPA_BLOFF)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | BLOFF[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

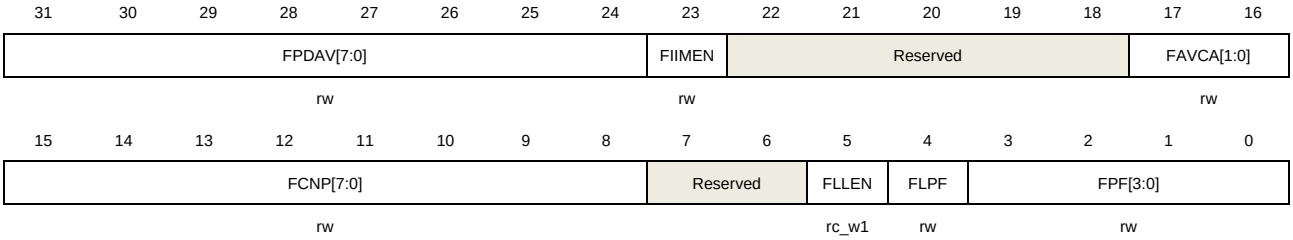
| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:0 | BLOFF[13:0] | Background line offset<br>These bits indicate the number of pixel between the last pixel of the current line and the first pixel of the next line. If the background pixel format is A4 or L4, the BLOFF must be configured to be an even number, otherwise a configuration error will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

## 35.7.8. Foreground pixel control register (IPA_FPCTL)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | FPDAV[7:0] | Foreground pre-defined alpha value<br>Software set and clear<br>These bits define an alpha value. These bits are used to calculate the foreground alpha channel value with the alpha value read from foreground memory or foreground LUT according to the foreground alpha calculation algorithm.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 23 | FIIMEN | Foreground input interlace mode enable<br>0: Foreground input interlace mode disable<br>1: Foreground input interlace mode enable |
| 22:18 | Reserved | Must be kept at reset value. |
| 17:16 | FAVCA[1:0] | Foreground alpha value calculation algorithm<br>Software set and clear<br>00: No effect<br>01: FPDAV[7:0] is selected as the foreground alpha value<br>10: FPDAV[7:0] multiplied by the alpha data read from foreground memory or foreground LUT divided by 255 is selected as the foreground alpha value<br>11: Reserved<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 15:8 | FCNP[7:0] | Foreground LUT number of pixel<br>Software set and clear<br>The pixel number of foreground LUT is equal to FCNP + 1.<br>These bits can NOT be written when FLLEN is '1'. |
| 7:6 | Reserved | Must be kept at reset value |
| 5 | FLLEN | Foreground LUT loading enable<br>Software set, hardware clear.<br>0: Disable foreground LUT loading<br>1: Enable foreground LUT loading<br>When this bit is enabled, the foreground LUT loading is started. This bit is automatically cleared when one of the following situations occurs:<br>- When the TST bit is enabled<br>- When the foreground LUT loading is finished<br>- When a wrong configuration or a transfer error is detected |

- When the IPA transfer is ongoing or the background LUT is being loaded (TEN bit in the IPA_CTL register or BLLEN bit in the IPA_BPCTL register is ‘1’).

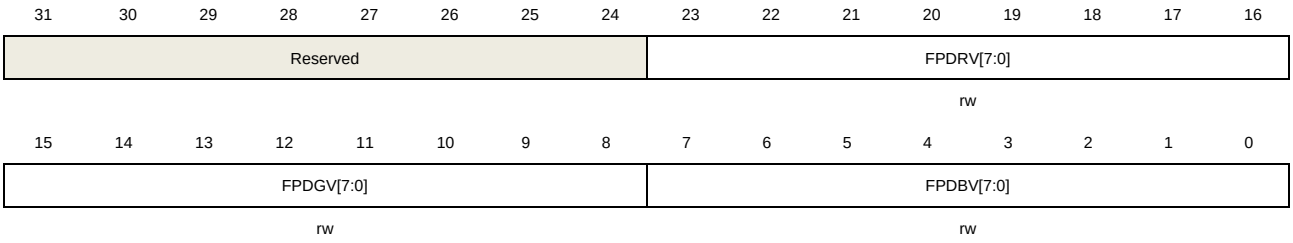
| | | |
|---|---|---|
| 4 | FLPF | Foreground LUT pixel format<br>Software set and clear<br>0: ARGB8888<br>1: RGB888<br>This bit can NOT be written when FLLEN is ‘1’. |
| 3:0 | FPF[3:0] | Foreground pixel format<br>software set and clear<br>0000: ARGB8888<br>0001: RGB888<br>0010: RGB565<br>0011: ARGB1555<br>0100: ARGB4444<br>0101: L8<br>0110: AL44<br>0111: AL88<br>1000: L4<br>1001: A8<br>1010: A4<br>1011: YUV444 (1 Plane)<br>1100: UYVY422 (1 Plane)<br>1101: VYUY422 (1 Plane)<br>1110: YUV420 (2 Plane)<br>1111: YVU420 (2 Plane)<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

## 35.7.9. Foreground pixel value register (IPA_FPV)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | FPDRV[7:0] | Foreground pre-defined red value |

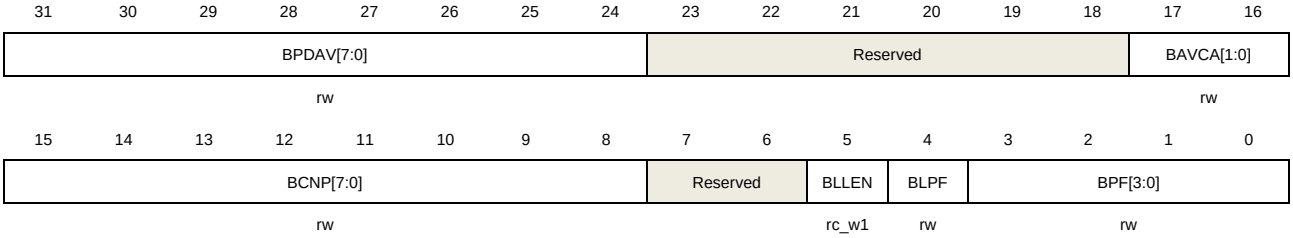
| | | |
|---|---|---|
| | | Software set and clear<br>When the foreground pixel format is A4 or A8, these bits are used as the foreground red value.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 15:8 | FPDGV[7:0] | Foreground pre-defined green value<br>Software set and clear<br>When the foreground pixel format is A4 or A8, these bits are used as the foreground green value.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 7:0 | FPDBV[7:0] | Foreground pre-defined blue value<br>Software set and clear<br>When the foreground pixel format is A4 or A8, these bits are used as the foreground blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

## 35.7.10. Background pixel control register (IPA_BPCTL)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | BPDAV[7:0] | Background pre- defined alpha value<br>Software set and clear<br>These bits define an alpha value. These bits are used to calculate the background alpha channel value with the alpha value read from background memory or background LUT according to the background alpha calculation algorithm.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 23:18 | Reserved | Must be kept at reset value. |
| 17:16 | BAVCA[1:0] | Background alpha value calculation algorithm<br>Software set and clear<br>00: No effect<br>01: BPDAV[7:0] is selected as the background alpha value<br>10: BPDAV[7:0] multiplied by the alpha data read from background memory or background LUT divided by 255 is selected as the background alpha value |

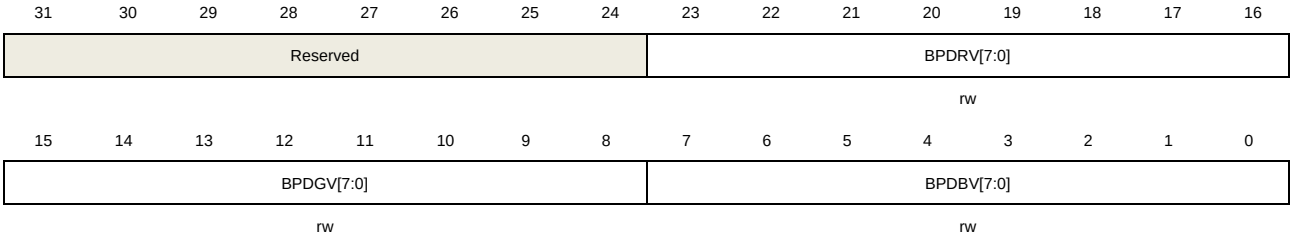
| | | |
|---|---|---|
| | | 11: Reserved<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 15:8 | BCNP[7:0] | Background LUT number of pixel<br>Software set and clear<br>The number of pixel of background LUT is equal to BCNP + 1.<br>These bits can NOT be written when BLLEN is ‘1’. |
| 7:6 | Reserved | Must be kept at reset value |
| 5 | BLLEN | Background LUT loading enable<br>Software set, hardware clear.<br>0: Background LUT loading disable<br>1: Background LUT loading enable<br>This bit is automatically cleared when one of the following situations occurs:<br>- When the TST bit is enabled<br>- When the background LUT loading is finished<br>- When a wrong configuration or a transfer error is detected<br>- When the IPA transfer is ongoing or the foreground LUT is being loaded (TEN bit in the IPA_CTL register or FLLEN bit in the IPA_FPCTL register is ‘1’). |
| 4 | BLPF | Background LUT pixel format<br>Software set and clear<br>0: ARGB8888<br>1: RGB888<br>This bit can NOT be written when BLLEN is ‘1’. |
| 3:0 | BPF[3:0] | Background pixel format<br>software set and clear<br>0000: ARGB8888<br>0001: RGB888<br>0010: RGB565<br>0011: ARGB1555<br>0100: ARGB4444<br>0101: L8<br>0110: AL44<br>0111: AL88<br>1000: L4<br>1001: A8<br>1010: A4<br>1011 ~ 1111: Reserved<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

### 35.7.11. Background pixel value register (IPA_BPV)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



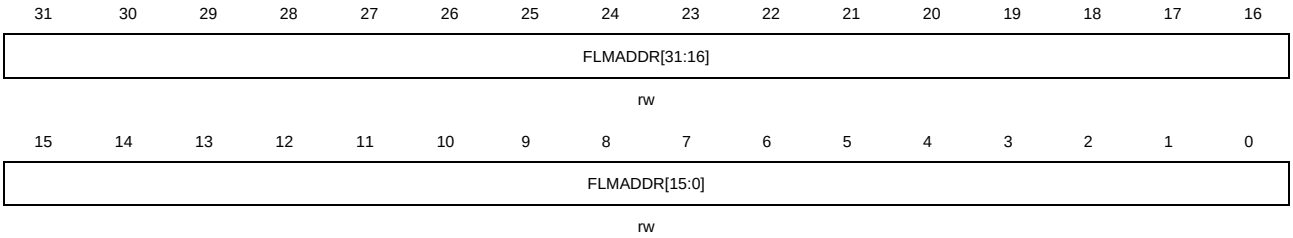

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | BPDRV[7:0] | Background pre-defined red value<br>Software set and clear<br>When the background pixel format is A4 or A8, these bits are used as the background red value.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 15:8 | BPDGV[7:0] | Background pre-defined green value<br>Software set and clear<br>When the background pixel format is A4 or A8, these bits are used as the background green value.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 7:0 | BPDBV[7:0] | Background pre-defined blue value<br>Software set and clear<br>When the background pixel format is A4 or A8, these bits are used as the background blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

## 35.7.12. Foreground LUT memory base address register (IPA_FLMADDR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



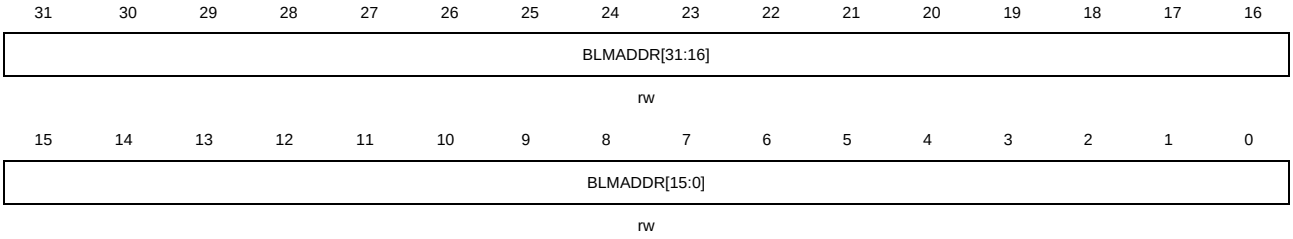

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FLMADDR[31:0] | Foreground LUT memory base address<br>Software set and clear |

The address must be aligned to 8-bit, 16-bit or 32-bit corresponding with the foreground LUT pixel format. If foreground LUT pixel format is ARGB8888, these bits must be 32-bit aligned. If the above alignment rule is broken, a wrong configuration will be detected when the foreground LUT loading is enable.
These bit can NOT be written when FLLEN in the IPA_FPCTL register is '1'.

## 35.7.13. Background LUT memory base address register (IPA_BLMADDR)

Address offset: 0x30
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

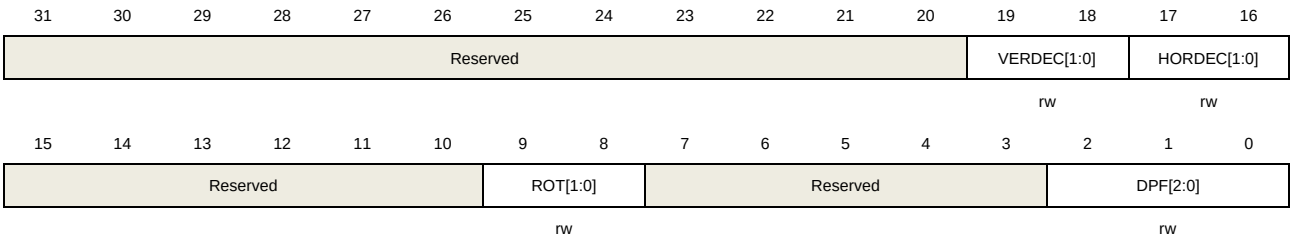



| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | BLMADDR[31:0] | Background LUT memory base address<br>Software set and clear<br>The address must be aligned to 8-bit, 16-bit or 32-bit corresponding with the background LUT pixel format. If background LUT pixel format is ARGB8888, these bits must be 32-bit aligned. If the above alignment rule is broken, a wrong configuration will be detected when the background LUT loading is enable.<br>These bit can NOT be written when BLLEN in the IPA_BPCTL register is '1'. |

## 35.7.14. Destination pixel control register (IPA_DPCTL)

Address offset: 0x34
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value. |
| 19:18 | VERDEC[1:0] | Verticle pre decimation filter control |

| | | |
|---|---|---|
| | | 00: Disable the pre-decimation filter<br>01: Down scaling factors of decimation filter is 2<br>10: Down scaling factors of decimation filter is 4<br>11: Down scaling factors of decimation filter is 8 |
| 17:16 | HORDEC[1:0] | Horizontal pre decimation filter control<br>00: Disable the pre-decimation filter<br>01: Down scaling factors of decimation filter is 2<br>10: Down scaling factors of decimation filter is 4<br>11: Down scaling factors of decimation filter is 8 |
| 15:10 | Reserved | Must be kept at reset value. |
| 9:8 | ROT[1:0] | Indicates the clockwise rotation to be applied at the output.<br>00: No Rotation<br>01: Rotate 90 degree<br>10: Rotate 180 degree<br>11: Rotate 270 degree |
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | DPF[2:0] | Destination pixel format<br>Software set and clear<br>000: ARGB8888<br>001: RGB888<br>010: RGB565<br>011: ARGB1555<br>100: ARGB4444<br>101~111: Reserved<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

## 35.7.15. Destination pixel value register (IPA_DPV)

Address offset: 0x38

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DPDAV[7:0] | | | | | | | | DPDRV[7:0] | | | | | | | |
| MEANINGLESS | | | | | | | | DPDRV[7:0] | | | | | | | |
| MEANINGLESS | | | | | | | | | | | | | | | |
| MEANINGLESS | | | | | | | | | | | | | | | |
| MEANINGLESS | | | | | | | | | | | | | | | |

rw

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DPDGV[7:0] | | | | | | | | DPDBV[7:0] | | | | | | | |
| DPDGV[7:0] | | | | | | | | DPDBV[7:0] | | | | | | | |

| | | | |
|---|---|---|---|
| DPDRV[4:0] | | DPDGV[5:0] | DPDBV[4:0] |
| DPDAV | DPDRV[4:0] | DPDGV[4:0] | DPDBV[4:0] |
| DPDAV[3:0] | DPDRV[3:0] | DPDGV[3:0] | DPDBV[3:0] |

rw

**When the destination pixel format is ARGB8888, the FIRST row is valid.**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DPDAV[7:0] | Destination pre-defined alpha value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination alpha value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 23:16 | DPDRV[7:0] | Destination pre-defined red value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination red value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 15:8 | DPDGV[7:0] | Destination pre-defined green value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination green value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 7:0 | DPDBV[7:0] | Destination pre-defined blue value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

**When the destination pixel format is RGB888, the SECOND row is valid.**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Meaningless | These bit can be set and cleared by software, but these bits have no meaning when the destination pixel format is RGB888. |
| 23:16 | DPDRV[7:0] | Destination pre-defined red value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination red value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 15:8 | DPDGV[7:0] | Destination pre-defined green value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination green value. |

| | | |
|---|---|---|
| | | These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 7:0 | DPDBV[7:0] | Destination pre-defined blue value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

**When the destination pixel format is RGB565, the THIRD row is valid.**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Meaningless | These bit can be set and cleared by software, but these bits have no meaning when the destination pixel format is RGB565. |
| 15:11 | DPDRV[4:0] | Destination pre-defined red value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination red value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 10:5 | DPDGV[5:0] | Destination pre-defined green value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination green value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 4:0 | DPDBV[4:0] | Destination pre-defined blue value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

**When the destination pixel format is ARGB1555, the FOURTH row is valid.**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Meaningless | These bit can be set and cleared by software, but these bits have no meaning when the destination pixel format is ARGB1555. |
| 15 | DPDAV | Destination pre-defined alpha value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination alpha value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 14:10 | DPDRV[4:0] | Destination pre-defined red value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination red value. |

| | | |
|---|---|---|
| | | These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 9:5 | DPDGV[4:0] | Destination pre-defined green value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination green value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 4:0 | DPDBV[4:0] | Destination pre-defined blue value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

**When the destination pixel format is ARGB4444, the FIFTH row is valid.**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Meaningless | These bit can be set and cleared by software, but these bits have no meaning when the destination pixel format is ARGB4444. |
| 15:12 | DPDAV[3:0] | Destination pre-defined alpha value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination alpha value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 11:8 | DPDRV[3:0] | Destination pre-defined red value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination red value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 7:4 | DPDGV[3:0] | Destination pre-defined green value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination green value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |
| 3:0 | DPDBV[3:0] | Destination pre-defined blue value<br>Software set and clear<br>When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

### 35.7.16. Destination memory base address register (IPA_DMADDR)

Address offset: 0x3C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DMADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMADDR[31:0] | Destination memory base address<br>software set and clear<br>The address must be aligned to 8-bit, 16-bit or 32-bit corresponding with the destination pixel format. If the destination pixel format is ARGB8888, these bits must be 32-bit aligned; If the destination pixel format is RGB565, ARGB1555 or ARGB4444, these bits must be 16-bit aligned. If the above alignment rules are broken, a wrong configuration will be detected when the transfer is enable.<br>These bit can NOT be written when TEN in the IPA_CTL register is ‘1’. |

## 35.7.17. Destination line offset register (IPA_DLOFF)

Address offset: 0x40

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DLOFF[13:0] | | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

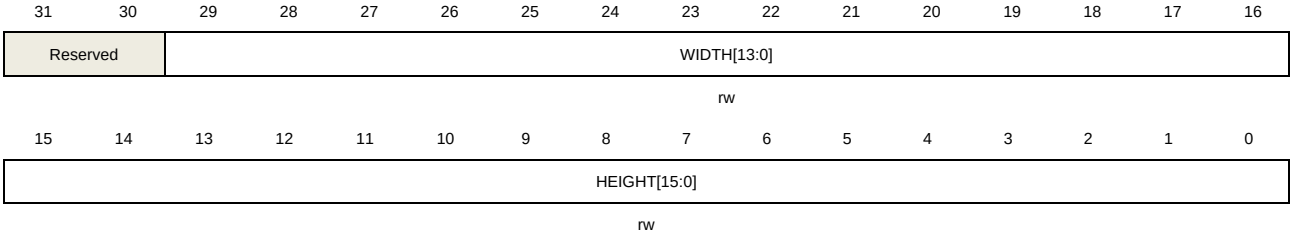
| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:0 | DLOFF[13:0] | Destination line offset<br>software set and clear<br>These bits indicate the number of pixel between the last pixel of the current line and the first pixel of the next line. When PFCM in the IPA_CTL register is configured to ‘00’, if the foreground pixel format is A4 or L4, these bits must be configured to be an even number, or a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

### 35.7.18. Image size register (IPA_IMS)

Address offset: 0x44

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



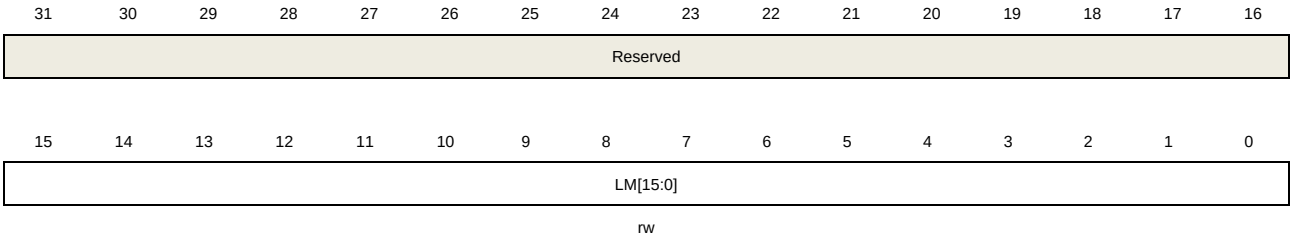

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:16 | WIDTH[13:0] | Width of the image to be processed<br>Software set and clear<br>These bits signify the number of pixels per line. If the foreground or background pixel format is A4 or L4, these bits must be configured to be an even number, otherwise a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |
| 15:0 | HEIGHT[15:0] | Height of the image to be processed<br>Software set and clear<br>These bits specify the number of lines of image to be processed.<br>These bits can NOT be written when TEN in the IPA_CTL register is ‘1’. |

### 35.7.19. Line mark register (IPA_LM)

Address offset: 0x48

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



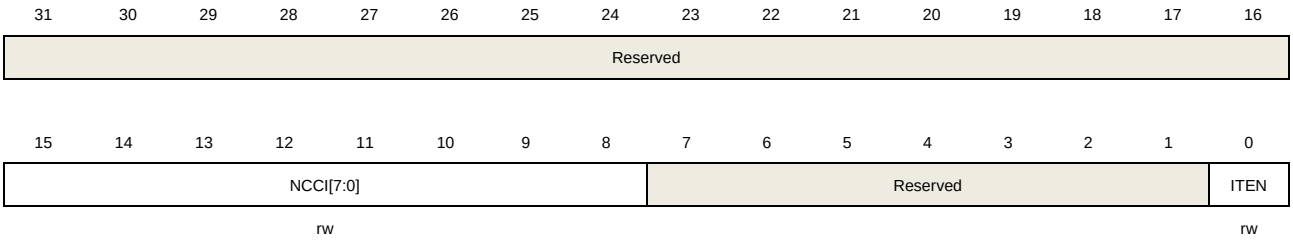

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | LM[15:0] | line mark<br>Software set and clear |

These bits define a line number to signify the transfer level. An interrupt flag is asserted as soon as the last pixel of the marked line has been written into the destination memory.
These bits can NOT be written when TEN in the IPA_CTL register is '1'.

## 35.7.20. Inter-timer control register (IPA_ITCTL)

Address offset: 0x4C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

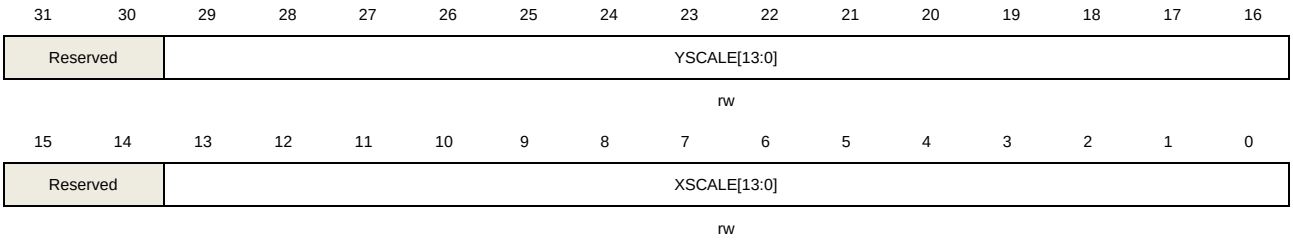



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | NCCI[7:0] | Number of clock cycles interval<br>Software set and clear<br>These bits have no meaning if ITEN is '0'. If the ITEN is '1', two consecutive AHB commands are issued with an interval equal to or greater than these bits. |
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | ITEN | Inter-timer enable<br>An inter-timer is implemented to reduce the AHB bus bandwidth usage of IPA.<br>0: Disable inter-timer<br>1: Enable inter-timer |

## 35.7.21. Bilinear scaling control register (IPA_BSCTL)

Address offset: 0x50
Reset value: 0x1000 1000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).



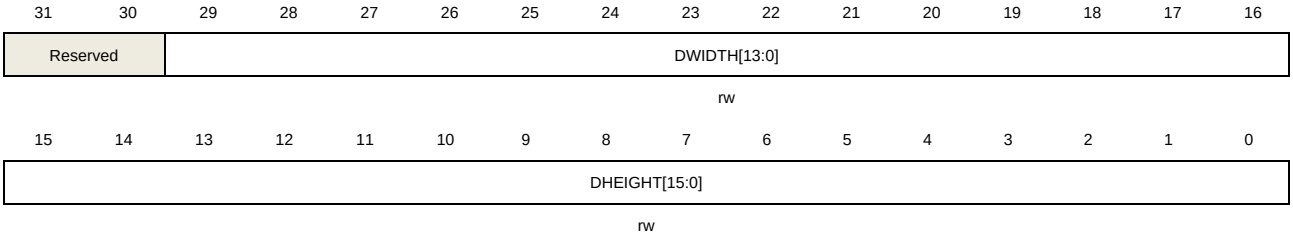
| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:16 | YSCALE[13:0] | It consists of 2-bit integer and 12-bit fractional, representing the Y scaling factor for the Foreground.<br>**Note:** The maximum down scaling factor is 1/2 and the maximum up scaling factor is 2^12. |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:0 | XSCALE[13:0] | It consists of 2-bit integer and 12-bit fractional, representing of the X scaling factor for the Foreground.<br>**Note:** The maximum down scaling factor is 1/2 and the maximum up scaling factor is 2^12. |

## 35.7.22. Scaling destination image size register (IPA_DIMS)

Address offset: 0x54

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:16 | DWIDTH[13:0] | Width of the image to be processed<br>Software set and clear.<br>These bits signify the number of pixels per line. If the foreground or background pixel format is A4 or L4, these bits must be configured to be an even number, otherwise a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'.<br>Set these bits when scaling. |
| 15:0 | DHEIGHT[15:0] | Height of the image to be processed<br>Software set and clear<br>These bits specify the number of lines of image to be processed.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'.<br>Set these bits when scaling. |

### 35.7.23. Foreground even frame/UV memory base address register (IPA_EF_UV_MADDR)

Address offset: 0x5C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EFUVMADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EFUVMADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | EFUVMADDR[31:0] | Foreground Even Frame/UV memory base address<br>These bits must be aligned to 8-bit, 16-bit or 32-bit corresponding with the foreground pixel format. If foreground pixel format is ARGB8888,UYVY422 or VYUY422, these bits must be 32-bit aligned; If the foreground pixel format is YUV420, YVU420, RGB565, ARGB1555, ARGB4444 or AL88, these bits must be 16-bit aligned. If the above alignment rules are broken, a wrong configuration will be detected when the transfer is enable.<br>These bits can NOT be written when TEN in the IPA_CTL register is '1'. |

### 35.7.24. Color space conversion coefficient configure register 0 (IPA_CSCC_CFG0)

Address offset: 0x60

Reset value: 0x0400 0000

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CONVMOD | Reserved | | | | | | | C0[10:0] | | | | | | UVOFF[8:7] | |
| rw | | | | | | | | rw | | | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UVOFF[6:0] | | | | | | | YOFF[8:0] | | | | | | | | |
| rw | | | | | | | rw | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CONVMOD | Color space convert mode |

| | | |
|---|---|---|
| | | 0: Convert YUV to RGB, the UV value are 8-bit two's compliment<br>1: Convert YCbCr to RGB, the CbCr value are 8-bit unsigned number |
| 30:29 | Reserved | Must be kept at reset value. |
| 28:18 | C0[10:0] | Y multiplier coefficient. For YUV, it is typically 0x100 (1.000). For YCbCr, it is typically 0x12A (1.164). |
| 17:9 | UVOFF[8:0] | Phase offset implicited for UV/CbCr data, which is used for YUV/YCbCr to RGB conversion. For YUV, it is typically 0x000. For YCbCr, it is typically 0x180 (-128). |
| 8:0 | YOFF[8:0] | Amplitude offset implicited for Y data. For YUV, it is typically 0. For YCbCr, it is typically 0x1F0 (-16). |

### 35.7.25. Color space conversion coefficient configure register 1 (IPA_CSCC_CFG1)

Address offset: 0x64
Reset value: 0x0123 0208

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | C1[10:0] | | | | | | | | | | |
| | | | | | rw | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | C4[10:0] | | | | | | | | | | |
| | | | | | rw | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26:16 | C1[10:0] | V/Cr Red multiplier coefficient. For YUV, it is typically 0x123 (1.140). For YCbCr it is typically 0x198 (1.596). |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | C4[10:0] | U/Cb Blue multiplier coefficient. For YUV, it is typically 0x208 (2.032). For YCbCr it is typically 0x204 (2.017). |

### 35.7.26. Color space conversion coefficient configure register 2 (IPA_CSCC_CFG2)

Address offset: 0x68
Reset value: 0x076B 079C

This register can be accessed by byte (8-bit), half-word (16-bit) and word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|




| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26:16 | C2[10:0] | V/Cr Green multiplier coefficient. For YUV, it is typically 0x76B (-0.581). For YCbCr, it is typically 0x730 (-0.813). |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:0 | C3[10:0] | U/Cb Green multiplier coefficient. For YUV, it is typically 0x79C (-0.394). For YCbCr, it is typically 0x79C (-0.392). |

# 36. Secure digital input/output interface (SDIO) (For GD32H737_757_759)

## 36.1. Overview

The secure digital input/output interface (SDIO) defines the SD, SD I/O and embedded MultiMediaCard (*e*•MMC) host interface, which provides command/data transfer between the AHB system bus and SD memory cards, SD I/O cards and *e*•MMC.

The supported SD memory card and SD I/O card system specifications are defined in the SD card Association website at ***www.sdcard.org***.

The supported embedded Multimedia Card system specifications are defined through the Multimedia Card Association website at ***www.jedec.org***, published by the JEDEC SOLID STATE TECHNOLOGY ASSOCIATION.

## 36.2. Characteristics

The SDIO features include the following:

- ***e*•MMC:** Support for embedded Multimedia Card System Specification Version 4.51 (and previous versions) Card and five different data bus modes: 1-bit (default), 4-bit (SDR/DDR) and 8-bit(SDR/DDR).
- **SD Card:** Full support for *SD Memory Card Specifications Version 3.0.*
- **SD I/O:** Full support for *SD I/O Card Specification Version 3.0* card and three different data bus modes: 1-bit (default) and 4-bit (SDR/DDR).
- 104MHz data transfer frequency and 8-bit data transfer mode.
- Interrupt and DMA request to processor.
- Support DDR double data rate signaling.

**Note:** SDIO supports only one SD card, SD I/O card or *e*•MMC at any one time and a stack of e•MMC V4.51 or previous.

There are two SDIO interfaces, and the implementation is shown in ***Table 36-1. 2 SDIOs***.

**Table 36-1. 2 SDIOs**

| SDIO features | SDIO0 | SDIO1 |
|---|---|---|
| Delay module (SDR104, HS200) | + | + |
| SDIO_CLKIN | + | - |
| SDIO_CMDDIR, SDIO_DAT0DIR, SDIO_DAT123DIR | + | - |
| MDMA command end, MDMA data transfer end, MDMA buffer end | + | - |

**Note:** "-" means no support; "+" means support.

## 36.3. SDIO function overview

### 36.3.1. SDIO bus topology

After a power-on reset, the host must initialize the card by a special message-based bus protocol.

Each message is represented by one of the following tokens:

**Command**: a command is a token which starts an operation. A command is sent from the host to a card. A command is transferred serially on the CMD line.

**Response**: a response is a token which is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

**Data**: data can be transferred from the card to the host or vice versa. Data is transferred via the data lines. The number of data lines used for the data transfer can be 1(DAT0), 4(DAT0-DAT3) or 8(DAT0-DAT7).

The structure of commands, responses and data blocks is described in ***SDIO function overview***. One data transfer is a bus operation.

There are different types of operations. Operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. The bits on the DAT0-DAT7 and CMD lines are transferred synchronous to the host clock.

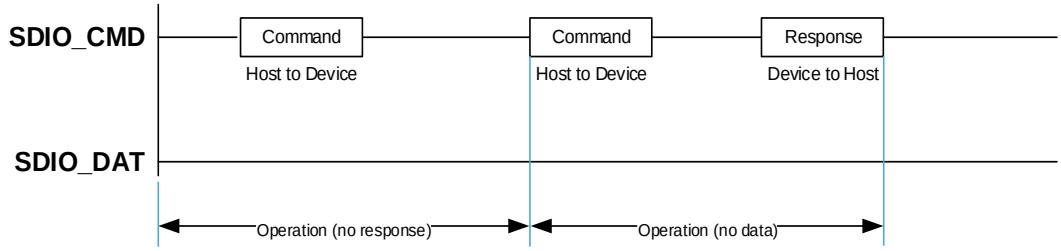
Three types of data transfer commands are defined:

- Stream commands: These commands initiate a continuous data stream; they are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum. (only *e*•MMC supports)
- Block-oriented commands: These commands send a data block successfully by CRC bits. Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.
- Multibyte mode: Single data block with block size range 1-512 bytes. (only SD/SD I/O supports)

The basic transaction on the bus is the command/response transaction (refer to ***Figure 36-1. SDIO "no response" and "no data" operations***). This type of bus transaction transfers their information directly within the command or response structure. In addition, some operations

have a data token. Data transfers to/from the Card are done in blocks.

**Figure 36-1. SDIO “no response” and “no data” operations**




Note that the Multiple Block operation mode is faster than Single Block operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the host to use single or multiple data lines. ***Figure 36-2. SDIO multiple blocks read operation*** is the multiple blocks read operation and ***Figure 36-3. SDIO multiple blocks write operation*** is the multiple block write operation. The block write operation uses a simple busy signal of the write operation duration on the data (DAT0) line.

**Figure 36-2. SDIO multiple blocks read operation**

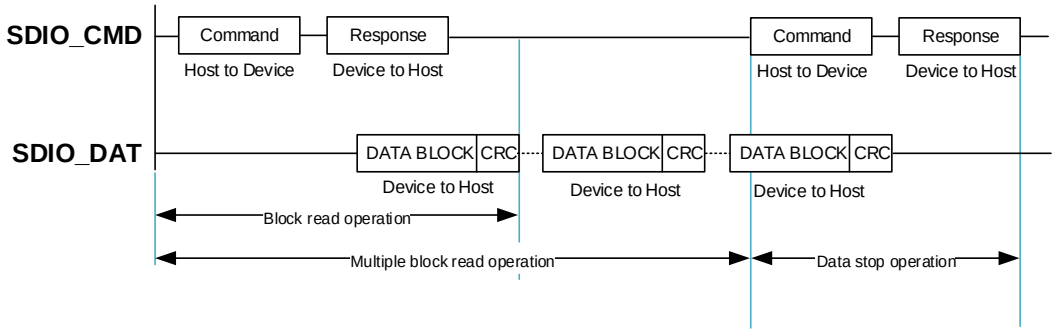
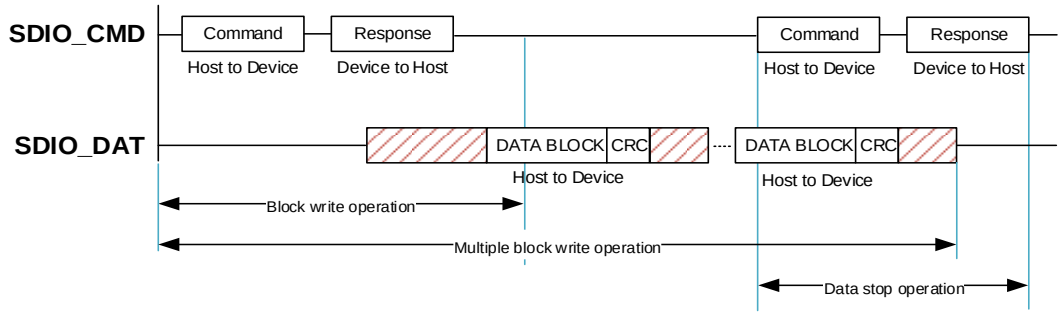



**Figure 36-3. SDIO multiple blocks write operation**




Data transfers to/from SD memory cards and SD I/O cards (both IO only card and combo card) are done in data blocks. Data transfers to/from e•MMC are done in data blocks or streams. ***Figure 36-4. SDIO sequential read operation*** and ***Figure 36-5. SDIO sequential***

***write operation*** are the stream read and write operation.

**Figure 36-4. SDIO sequential read operation**

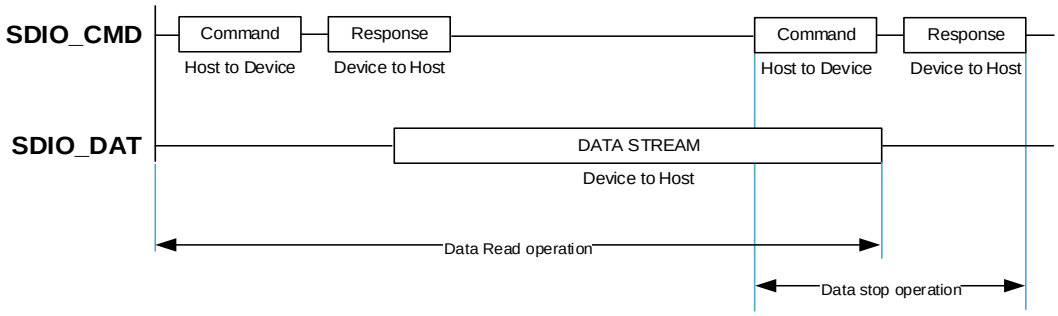
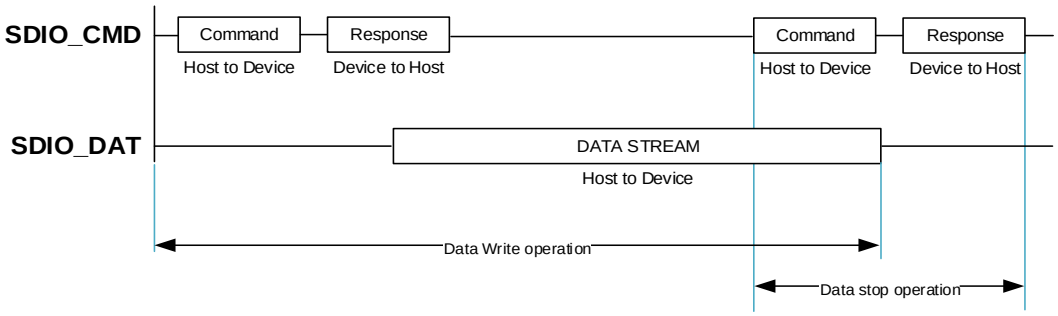



**Figure 36-5. SDIO sequential write operation**




## 36.3.2. SDIO operation modes

**Table 36-2. SDIO operation modes for SD & SD I/O card**

| Bus Speed modes | Max Bus Speed [Mbyte/s] | Max Clock frequency [MHz] | Signal Voltage [V] |
|---|---|---|---|
| DS | 12.5 | 25 | 3.3 |
| HS | 25 | 50 | 3.3 |
| SDR12 | 12.5 | 25 | 1.8 |
| SDR25 | 25 | 50 | 1.8 |
| DDR50 | 50 | 50 | 1.8 |
| SDR50 | 50 | 100 | 1.8 |
| SDR104 | 104 | 208 | 1.8 |

**Table 36-3. SDIO operation modes for *e*•MMC card**

| Bus Speed modes | Max Bus Speed [Mbyte/s] | Max Clock frequency [MHz] | Signal Voltage [V] |
|---|---|---|---|
| Backward compatible | 26 | 26 | 3/1.8 |
| HS SDR | 52 | 52 | 3/1.8 |
| HS DDR | 104 | 52 | 3/1.8 |
| HS200 | 200 | 200 | 1.8 |

**Note:** 1. DS (Default Speed), HS (High Speed). 2. SDR (single data rate signaling), DDR (double data rate signaling). 3. The SD/SD I/O card max bus speed with 4-bit bus width, and the *e*•MMC max bus speed with 8-bit bus width. 4. Maximum frequency depending on

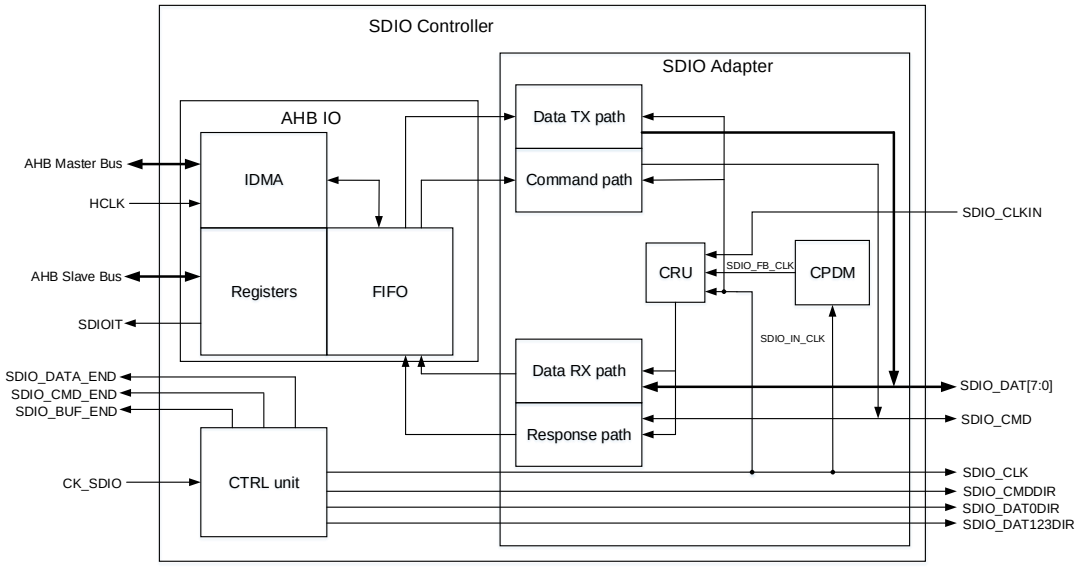
maximum allowed input/output speed. 5. The SDR104 and HS200 mode requires Delay module (CPDM) support using sampling point tuning. The use of Delay module for SDR50 and DDR50 mode is optional.

### 36.3.3. Block diagram

The following figure *Figure 36-6. SDIO block diagram* shows the SDIO structure. There have three main parts:

- The SDIO adapter block consists of control unit which manage clock, command unit which manage command transfer, data unit which manage data transfer.
- The AHB slave interface block contains access registers by AHB bus, contains FIFO unit which is data FIFO used for data transfer, and generates interrupt and DMA request signals.
- The internal DMA (IDMA) block with its AHB master interface.

**Figure 36-6. SDIO block diagram**




### 36.3.4. SDIO pins and internal signals

**Table 36-4. SDIO internal input/output singnals**

| Signal name | Signal type | Description |
|---|---|---|
| CK_SDIO | I | SDIO kernel clock |
| HCLK | I | AHB clock |
| SDIOIT | O | SDIO global interrupt |
| SDIO_CMD_END | O | SDIO command end trigger for MDMA |
| SDIO_DATA_END | O | SDIO data end trigger for MDMA |
| SDIO_BUF_END | O | SDIO internal DMA buffer end trigger for MDMA |
| SDIO_IN_CLK | I | The card feedback clock signal is connected to the SDIO_CLK pin (for DS/HS modes). |
| SDIO_FB_CLK | I | The card tuned feedback clock signal after CPDM (for DDR50, |

| Signal name | Signal type | Description |
|---|---|---|
| | | SDR50, SDR104, HS200). |

**Table 36-5. SDIO pins functions**

| Pin name | Signal name | Signal type | Description |
|---|---|---|---|
| SDIO_CK | SDIO_CLK | O | Clock to SD/SD I/O/e•MMC card |
| SDIO_CKIN | SDIO_CLKIN | I | Clock feedback from an external driver for card. (for SDR12, SDR25, SDR50, DDR50) |
| SDIO_CMD | SDIO_CMD | I/O | Bidirectional command/response signal |
| SDIO_CMDDIR | SDIO_CMDDIR | O | Direction indication for the SDIO_CMD signal |
| SDIO_D[7:0] | SDIO_DAT[7:0] | I/O | Data input/output for data lines DAT[7:0] |
| SDIO_DAT0DIR | SDIO_DAT0DIR | O | Direction indication for the SDIO_DAT0 data line |
| SDIO_DAT123DIR | SDIO_DAT123DIR | O | Direction indication for the data lines SDIO_DAT[3:1] |

**Note:** I: input; O: output.

### 36.3.5. General description

The SDIO_DAT[7:0] lines have different operating modes:

- When the card is powered on, it uses 1-bit (SDIO_DAT0) data bus by default, and can be modified by register later.
- For an SD or an SD I/O card, 1-bit (SDIO_DAT0) or 4-bit (SDIO_DAT [3:0]) can be used. All data lines operate in push-pull mode.
- For an e•MMC, 1-bit (SDIO_DAT0), 4-bit (SDIO_DAT[3:0]) or 8-bit (SDIO_DAT[7:0]) data bus widths can be used.

To allow the connection of a voltage switch transceiver, the I/O direction signals are used to indicates the direction of data flow on the data lines. The SDIO_DAT0DIR signal indicates the I/O direction for the SDIO_DAT0 data line, the SDIO_DAT123DIR for the SDIO_DAT[3:1] data lines, and the direction of data flow on the SDIO_CMD line is indicated with the I/O direction signal SDIO_CMDDIR. SDIO_CMD only operates in push-pull mode.

SDIO_CLK clock to the card originates from CK_SDIO:

- If the CK_SDIO clock has 50% duty cycle, it can be used even DIV = 0.
- If the CK_SDIO duty cycle is not 50%, the DIV > 0 must satisfy.
- The phase relation between the SDIO_CMD/SDIO_DAT[7:0] outputs and the SDIO_CLK can be selected through the CLKEDGE bit. The phase relation depends on the DIV, CLKEDGE, and DRSEL settings. Refer to ***Table 36-6. SDIO CMD and data phase selection.***

**Table 36-6. SDIO CMD and data phase selection**

| DIV | DRSEL | CLKEDGE | SDIO_CLK | CMD out | Data out |
|---|---|---|---|---|---|
| = 0 | - | - | = CK_SDIO | generated on CK_SDIO falling edge | |
| > 0 | 0 | 0 | generated on CK_SDIO rising edge | generated on CK_SDIO falling edge succeeding the SDIO_CLK rising edge. | |
| > 0 | 0 | 1 | | generated on the same CK_SDIO rising edge that generates the SDIO_CLK falling edge. | |
| > 0 | 1 | 0 | generated on CK_SDIO rising edge | generated on CK_SDIO falling edge succeeding the SDIO_CLK rising edge. | generated on CK_SDIO falling edge succeeding a SDIO_CLK edge. |
| > 0 | 1 | 1 | | generated on the same CK_SDIO rising edge that generates the SDIO_CLK falling edge. | |

By default, SDIO select SDIO_IN_CLK feedback clock input samples the incoming data in the SDIO receiving path. The SDIO_IN_CLK is derived from the SDIO_CLK pin. For tuning the phase of the sampling clock to adapt to the timing of the received data, the CPDM on the device can be connected to the SDIO_FB_CLK clock signal and the SDIO_IN_CLK. Then select the input of SDIO_FB_CLK clock for the receive path to use the phase adjusted sampling clock for the incoming data. This is what SDR104/HS200 working mode must meet, and SDR50/DDR50 is optional.

The SDIO_CLKIN feedback clock input signal can be selected to sample the received data, when using an external driver (voltage switching transceiver).

For an SD /SD I/O/ *e*•MMC card, the clock frequency can vary from 0 to 208 MHz (limited by maximum input/output speed).

Depending on the selected bus mode (SDR or DDR), one or two bits are transferred on SDIO_DAT[7:0] lines per clock cycle. The SDIO_CMD line transfers only one bit per clock cycle.

### 36.3.6. SDIO adapter

The SDIO adapter contains control unit, command unit and data unit, and generates signals to cards. The signals is descript bellow:

**SDIO_CLK**: The SDIO_CLK is the clock provided to the card. Each cycle of this signal directs a one bit transfer on the command line (SDIO_CMD). The SDIO_CLK frequency can vary between 0 MHz and 48 MHz for a MMC Card V4.2, between 0 and 200 MHz for a *e*•MMC V4.51, or between 0 and 208 MHz for an SD/SD I/O card.

SDIO uses two clock signals: the SDIO adapter clock and the AHB bus clock (HCLK).

**SDIO_CMD**: This signal is a bidirectional command channel used for card initialization and transfer of commands. Commands are sent from the SDIO controller to the card and

responses are sent from the card to the host. The CMD signal has two operation modes: open-drain for initialization (used only for e•MMC card initialization), and push-pull for command transfer (SD/SD I/O card and data transfer for *e*•MMC card).

**SDIO_DAT[7:0]**: These are bidirectional data channels. The DAT signals operate in push-pull mode. Only the card or the host is driving these signals at a time. By default, after power up or reset, only DAT0 is used for data transfer. A wider data bus can be configured for data transfer, using either DAT0-DAT3 or DAT0-DAT7 (just for *e*•MMC 4.51), by the SDIO controller. The SDIO includes internal pull-ups for data lines DAT1-DAT7. Right after entering to the 4-bit mode the card disconnects the internal pull-ups of lines DAT1-DAT3. Correspondingly right after entering to the 8-bit mode the card disconnects the internal pull-ups of lines DAT1-DAT7.

**SDIO_CLKIN**: This signal is an digital input signal, and used for the clock feedback from an external driver of SD/SD I/O/*e*•MMC card. (for SDR12/SDR25/SDR50/DDR50)

**SDIO_CMDDIR**: This signal is an digital output signal. As card I/O direction indication for the SDIO_CMD signal.

**SDIO_DAT0DIR**: This signal is an digital output signal. As card I/O direction indication for the SDIO_DAT0 data line.

**SDIO_DAT123DIR**: This signal is an digital output signal. SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDIO_DAT[3:1].

The SDIO adapter is an interface to SD, SD I/O and *e*•MMC. It consists of several subunits:

### Control unit

The control unit contains the power management functions, the clock management functions for the memory card clock and I/Os direction management.

The power management is controlled by SDIO_PWRCTL register which implements power off or power on. The power management subunit will disable the card bus output signal during the reset, power-off, and power-up phases. The power saving mode configured by setting CLKPWRSAV bit in SDIO_CLKCTL register, which implements close the SDIO_CLK when the bus is idle.

Clock management uses CK_SDIO to generate the SDIO_CLK clock signal to the card and provide divider control. Adapter register and FIFO use AHB bus clock (HCLK). The control unit, command send path, and data send path use the SDIO adapter clock domain (CK_SDIO). The command response path and data receive path use the SDIO adapter feedback clock domain from SDIO_IN_CLK, SDIO_CLKIN, or SDIO_FB_CLK (generated by CPDM).

The I/O direction management controls the external voltage transceiver and controls signals for SDIO_CMDDIR, SDIO_D0DIR, and SDIO_D123DIR.

### Command unit

The command unit contains a command send path and a response receiv path, which are

used to send commands and receive responses to cards on the SDIO_CMD line. The command send path is clocked by SDIO_CLK, and the response receive channel has a dedicated SDIO internal receive clock. The data transfer flow is controlled by Command State Machine (CSM). After a write operation to SDIO_CMDCTL register and CSMEN in SDIO_CMDCTL register is 1, the command transfer starts. It firstly sends a command to the card. The command contains 48 bits send by SDIO_CMD signal which sends 1 bit to card at one SDIO_CLK. The 48 bits command contains 1 bit Start bit, 1 bit Transmission bit, 6 bits command index defined by CMDIDX bits in SDIO_CMDCTL register, 32 bits argument defined in SDIO_CMDAGMT register, 7 bits CRC, and 1 bit end bit. Then receive response from the card if CMDRESP in SDIO_CMDCTL register is not 0b00. There are short response which have 48 bits or long response which have 136 bits. The response stores in SDIO_RESP0 - SDIO_RESP3 registers. The command unit also generates the command status flags defined in SDIO_STAT register.

### Command state machine

| CS_Idle | After reset, ready to send command. | |
|---|---|---|
| 1.CSM enabled and WAITDEND enabled | → | CS_Pend |
| 2.CSM enabled, WAITDEND disabled and BOOTMODEN disabled | → | CS_Send |
| 3.CSM disabled | → | CS_Idle |
| 4.BOOTMODEN enabled | → | CS_Boot |
| **Note:** The state machine remains in the Idle state for at least eight SDIO_CLK periods to meet the $N_{CC}$ and $N_{RC}$ timing constraints. $N_{CC}$ is the minimum delay between two host commands, and $N_{RC}$ is the minimum delay between the host command and the response. | | |

| CS_Pend | Waits for the end of data transfer. | |
|---|---|---|
| 1.The data transfer complete | → | CS_Send |
| 2.CSM disabled | → | CS_Idle |
| **Note：** If DATALEN =< 5, CSM directly moves to the CS_Send state; If DATALEN > 5, CSM will wait for the signal from DSM before moves to the CS_Send state. | | |

| CS_Send | Sending the command. | |
|---|---|---|
| 1.The command transmitted has response | → | CS_Wait |
| 2.The command transmitted doesn't have response | → | CS_Idle |
| 3.CSM disabled | → | CS_Idle |

| CS_Wait | Wait for the start bit of the response. | |
|---|---|---|
| 1.Receive the response(detected the start bit) | → | CS_Receive |
| 2.Timeout is reached without receiving the response | → | CS_Idle |
| 3.CSM disabled | → | CS_Idle |
| **Note:** The command timeout has a fixed value of 64 SDIO_CLK clock periods. | | |

| CS_Receive | Receive the response and check the CRC. | |
|---|---|---|
| 1.CSM disabled | → | CS_Idle |
| 2.Response received | → | CS_Idle |
| 3.Command CRC failed | → | CS_Idle |

| CS_Boot | Read boot data from device. | |
|---|---|---|
| 1. BOOTMOD = 0 and BOOTMODEN enabled | → | CS_Boot |
| 2. BOOTMOD = 0 and BOOMODEN disabled | → | CS_Idle |
| 3.BOOTMOD = 1 | → | CS_Send |

### Data unit

The data unit performs data transfers to and from cards. The data transfer uses SDIO_DAT[7:0] signals when 8-bit data width (BUSMODE bits in SDIO_CLKCTL register is 0b10), use SDIO_DAT[3:0] signals when 4-bit data width (BUSMODE bits in SDIO_CLKCTL register is 0b01), or SDIO_DAT[0] signal when 1-bit data width (BUSMODE bits in SDIO_CLKCTL register is 0b00). The data transfer flow is controlled by Date State Machine (DSM). After a write operation to SDIO_DATACTL register and DATAEN in SDIO_DATACTL register is 1, the data transfer starts. It sends data to card when DATADIR in SDIO_DATACTL register is 0, or receive data from card when DATADIR in SDIO_DATACTL register is 1. The data unit also generates the data status flags defined in SDIO_STAT register. When data is received and the boot acknowledge is enabled, DSM move to the DS_WaitACK state and waits for the boot acknowledge, and then move to the DS_WaitR state.

### Data state machine

| DS_Idle | The data unit is inactive, waiting for send and receive. | |
|---|---|---|
| 1.(DSM enabled or DATAEN = 1) and (Not busy and data transfer direction is from host to card) | → | DS_WaitS |
| 2.DSM enabled and ACKEN disabled and data transfer direction is from card to host | → | DS_WaitR |
| 3.DSM enabled and ACKEN enabled and data transfer direction is from card to host | → | DS_WaitACK |

| DS_WaitS | Wait until the data FIFO empty flag is deasserted or data transfer ended. | |
|---|---|---|
| 1.Data transfer ended | → | DS_Idle |
| 2.DSM abort and data FIFO empty | → | DS_Idle |
| 3.DATHOLD flag is set | → | DS_Idle |
| 4.Data FIFO empty flag is deasserted and DATHOLD = 0 | → | DS_Send |

| DS_Send | Transmit data to the card. | |
|---|---|---|
| 1.Data block transmitted | → | DS_Busy |
| 2.DSM Abort | → | DS_Busy |
| 3.Data transfer ended | → | DS_Busy |

| DS_Busy | Waits for the CRC status flag. | |
|---|---|---|
| 1.Receive a positive CRC status | → | DS_WaitS |
| 2.No CRC and not busy | → | DS_WaitS |
| 3.Receive a negative CRC status and not busy | → | DS_Idle |
| 4.DSM disabled and not busy | → | DS_Idle |
| **Note:** The command timeout programmed in the data timer register (SDIO_DATATO). | | |

| DS_WaitR | Wait for the start bit of the receive data. | |
|---|---|---|
| 1.Data receive ended | → | DS_Idle |
| 2.DSM disabled | → | DS_Idle |
| 3. DATHOLD = 1 and FIFO empty. | → | DS_Idle |
| 4.Receives a start bit before timeout | → | DS_Receive |
| **Note:** The command timeout programmed in the data timer register (SDIO_DATATO). | | |

| DS_Receive | Receive data from the card and write it to the data FIFO. | |
|---|---|---|
| 1.Data block received and RWEN = 0 | → | DS_WaitR |
| 2.Data transfer ended and RWEN = 0 | → | DS_WaitR |
| 3.Data FIFO overrun error occurs | → | DS_Idle |
| 4.DSM disabled and FIFO empty. | → | DS_Idle |
| 5.CRC fails and data receive completed. | → | DS_Idle |

| DS_WaitACK | Wait for the boot acknowledgment token. | |
|---|---|---|
| 1.Receive the boot acknowledge in time and check OK | → | DS_WaitR |
| 2.Acknowledgement timed out or incorrect acknowledgement received | → | DS_WaitACK |
| 3.DSM disabled and FIFO empty | → | DS_Idle |

### CRU unit

The CRU unit selects the source for SDIO internal receive clock to be used with the received command response and data.

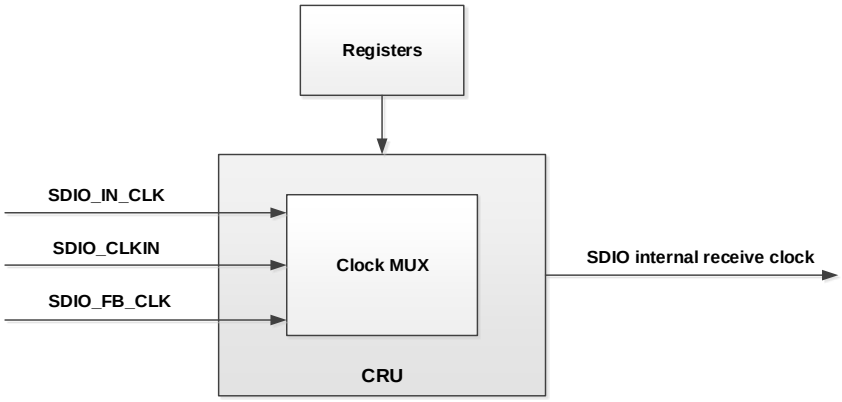
The RCLK[1:0] register can be set to select the clock source for receiving data, the three options are as follows:

- SDIO_IN_CLK bus master main feedback clock.
- SDIO_CLKIN external bus feedback clock.

- SDIO_FB_CLK bus tuned feedback clock.

**Note:** 1. When there is no external drive and DS/HS are used, select SDIO_IN_CLK. 2. Select SDIO_CLKIN when there are external drives and SDR12, SDR25, SDR50 and DDR50 are used. 3. When using CPDM, SDIO_FB_CLK must be selected input. If it is SDR104 or HS200, CPDM must be selected. If it is SDR50 or DDR50, CPDM is optional. 4. If CPM and DSM are in the idle state, the SDIO internal receive clock source must be changed.

**Figure 36-7. CRU unit**




## 36.3.7. AHB slave interface

The AHB slave interface implements access to SDIO registers, data FIFO and generates interrupt. It includes a data FIFO unit, registers unit, and the interrupt.

It consists the following subunits:

### Interrupts

The interrupt logic generates interrupt when at least one of the selected status flags is high. An interrupt enable register is provided to allow the logic to generate a corresponding interrupt. The status flag generates an interrupt request if the corresponding interrupt enable flag is set. Some status flags require an implicit clear in the clear register.

### Register unit

The register unit which contains all system registers generates the signals to control the communication between the controller and card.

### Data FIFO

The data FIFO unit has a data buffer, uses as transmit and receive FIFO. The FIFO contains a 32-bit wide, 32-word deep data buffer. When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

The transmit FIFO is used when write data to card. The data to be transferred is written to transmit FIFO by AHB bus, the data unit in SDIO adapter read data from transmit FIFO, and then send the data to card. When the DATALEN is not an integer multiple of 4, the last data

less than 4 bytes (1/2/3 bytes) will be written in word transfer.

The receive FIFO is used when read data from card. The data to be transferred is read from the card and then write to receive FIFO. The data in receive FIFO is read to AHB bus when needed. When the DATALEN is not an integer multiple of 4, the last data less than 4 bytes (1/2/3 bytes) are read with a word transfer padded with 0 value bytes.

### 36.3.8. AHB master interface

AHB main interface uses SDIO internal DMA (IDMA) to transfer data between FIFO and memory.

#### IDMA

IDMA provides a bidirectional high-speed data transmission channel between memory and FIFO.

The IDMA is enabled by setting the IDMAEN bit to 1, and supports 8-beat burst transmission.

Burst mode includes transmit mode and receive mode:

- Transmit of burst mode
  - As long as the FIFO is empty for the number of burst transfers, the data will be obtained from memory in the form of burst until all data indicated by DATALEN have been transmitted. If DATALEN is not a multiple of the burst size, the remaining data with less than the burst size will be transmitted in single transmission mode. When DATALEN is not a multiple of 4, the last data less than 4 bytes (1/2/3 bytes) is obtained by word transmission.
- Receive of burst mode
  - As long as the FIFO is empty for the number of burst transfers, the data will be stored in the memory in the form of burst until all the data indicated by DATALEN have been transmitted. If DATALEN is not a multiple of burst transmission, the remaining data with less than burst size will be transmitted in single transmission mode. When DATALEN is not a multiple of 4, the last data less than 4 bytes (1/2/3 bytes) are stored through half word or byte transmission.

In addition, IDMA also provides two channel configurations (selected by bit BUFMOD).

- Single buffered channel
  - In single buffer configuration, data on the memory side is accessed linearly from the base address IDMAADDR0. When IDMA completes the transmission of all data and DSM also completes the transmission, the DTEND flag will be set to 1.
- Double buffered channel
  - In the double buffer configuration, the data on the memory side is accessed from two buffers, one from IDMAADDR0 and the other from IDMAADDR1. In this way, when IDMA accesses one of the memory buffers, the firmware can process the other. The size of the memory buffer is defined by IDMASIZE. The buffer size should be an multiple of the burst size. When the channel is enabled, the base address of the buffer can be updated

immediately.

In the double buffered channel mode, the address of access memory is configured by the BUFSEL bit.

- When the BUFSEL bit is "0", IDMA hardware uses IDMAADDR0 to access the memory. When trying to write the register through firmware, the write operation will be discarded and IDMAADDR0 data will not change. Allow firmware write IDMAADDR1.
- When the BUFSEL bit is "1", IDMA hardware uses IDMAADDR1 to access memory. When an attempt is made to write the register through firmware, the write operation will be discarded and the IDMAADDR1 data will not change. Allow firmware to write IDMAADDR0.

When IDMA completes data transmission in one of the buffers, the buffer transmission completion flag (IDMAEND) will be set to 1 and the BUFSEL bit will flip, and then IDMA will continue to transmit data from the other buffer. When IDMA completes the transmission of all data and DSM also completes the transmission, the DTEND flag will be set to 1.

IDMAADDR0 and IDMAADDR1 addresses should be word aligned.

### 36.3.9. MDMA request

The internal trigger line (SDIO_DATA_END、SDIO_CMD_END and SDIO_BUF_END) from SDIO can send direct request to MDMA controller to realize continuous transmission from/to different internal RAM addresses without using CPU.

The request signal of MDMA is transmitted through SDIO_DATA_END pin input. At the same time, the signal input triggers the clearing of DTEND and CMDRECV flag, and finally starts a new transmission by directly accessing the SDIO control and configuration registers with MDMA, without CPU intervention.

When the response to the command is successfully received, the CMDRECV flag will be set. When the busy state after R1b response ends, DAT0BSY flag of status register is cleared and DAT0BSYEND flag is set. When the sequence command response associated with an eventual busy signal ends, set the SDIO_CMD_END output connected to the MDMA. So, the MDMA can manage CMD12 (STOP_TRANSMISSION) command (needed to support open mode transfers) by clearing CMDRECV and DAT0BSYEND status flags.

When using Linux operating system, the data to be transmitted through SDIO bus is contained in the 1-4kbyte blocks of the discontinuous address in the internal memory of the device. The double buffer mode allows the destination address of IDMA to be changed in internal memory. The IDMAEND flag of the status register is set each time the buffer transfer is completed. By signaling this event to MDMA through the SDIO_BUF_END output connected to an MDMA request input, the new buffer address base can alternatively fill the IDMAADDR0/IDMAADDR1 fields without CPU intervention.

The ***Table 36-7. SDIO connections to MDMA*** shows the actions of programming in MDMA according to SDIO requests:

**Table 36-7. SDIO connections to MDMA**

| Trigger signal | Event signaled | Event occurrence | MDMA transfer config | MDMA action |
|---|---|---|---|---|
| SDIO_DATA_END | End of successful data transfer | DTEND = 1 | single | Set DTENDC |
| SDIO_CMD_END | End of command sequence | CMDSEND = 1, or (CMDRECV = 1 and DAT0BSY = 0) | single | Set CMDSENDC<br>Set CMDRECVC<br>Set DAT0BSYENDC |
| SDIO_BUF_END | End of buffer reached | IDMAENDC = 1 | link list | Set IDMAENDC<br>Update IDMAADDR0/1 |

### 36.3.10. AHB and SDIO_CLK clock relation

AHB bandwidth should be at least 3 times of SDIO bus bandwidth.

**Table 36-8. AHB and e•MMC clock frequence relation**

| SDIO Bus modes | Bus width | Maximum SDIO_CLK [MHz] | Minimum AHB Clock [MHz] |
|---|---|---|---|
| DS | 8 | 26 | 19.5 |
| HS | 8 | 52 | 39 |
| HS DDR | 8 | 52 | 78 |
| HS200 | 8 | 200 | 150 |

**Table 36-9. AHB and SD/SD I/O card clock frequence relation**

| SDIO Bus modes | Bus width | Maximum SDIO_CLK [MHz] | Minimum AHB Clock [MHz] |
|---|---|---|---|
| SDR12 | 4 | 25 | 9.4 |
| SDR25 | 4 | 50 | 18.8 |
| SDR50 | 4 | 100 | 37.5 |
| DDR50 | 4 | 50 | 37.5 |
| SDR104 | 4 | 208 | 78 |

### 36.3.11. Hardware flow control

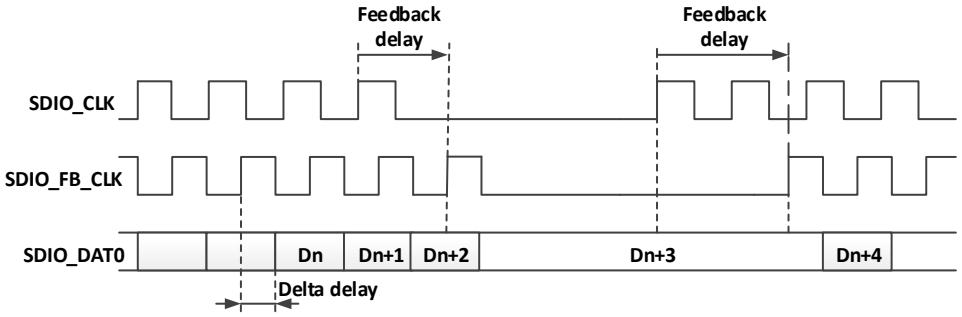
The hardware flow control function freezes the state machine by stopping the SDIO_CLK during data transmission, thereby preventing FIFO underrun (transmit data) and overrun (receive data) errors.

Stoppage of the FIFO causes a pause in data transmission until the transmit FIFO is half full or the DATALEN of the data is stored, or the receive FIFO is half empty. When the state machine freezes, the AHB interface may remain active. Therefore, the FIFO can operate even if hardware flow control is performed.

Hardware flow control is enabled by setting the HWEN register.

Hardware flow control must only be used when the SDIO_DAT data is cycle-aligned with the SDIO_CLK. SDR104 mode using CPDM cannot use hardware control.

**Figure 36-8. Hardware flow timing**




## 36.4. Card function overview

### 36.4.1. Card registers

Within the card interface registers are defined: OCR, CID, CSD, EXT_CSD, RCA, DSR and SCR. These can be accessed only by corresponding commands. The OCR, CID, CSD and SCR registers carry the card/content specific information, while the RCA and DSR registers are configuration registers storing actual configuration parameters. The EXT_CSD register carries both, card specific information and actual configuration parameters. For specific information, please refer to the relevant specifications.

**OCR register**: The 32-bit operation conditions register (OCR) stores the $V_{DD}$ voltage profile of the card and the access mode indication (*e*•MMC). In addition, this register includes a status information bit. This status bit is set if the card power up procedure has been finished. The register is a little different between *e*•MMC and SD card. The host can use CMD1 (e•MMC), ACMD41 (SD memory), CMD5 (SD I/O) to get the content of this register.

**CID register**: The Card Identification (CID) register is 128 bits wide. It contains the card identification information used during the card identification phase. Every individual Read/Write (RW) card shall have a unique identification number. The host can use CMD2 and CMD10 to get the content of this register.

**CSD register**: The Card-Specific Data register provides information regarding access to the card contents. The CSD defines the data format, error correction type, maximum data access time, data transfer speed, whether the DSR register can be used, etc. The programmable part of the register can be changed by CMD27. The host can use CMD9 to get the content of this register.

**Extended CSD Register**: Just *e*•MMC4.51 has this register. The Extended CSD register defines the card properties and selected modes. It is 512 bytes long. The most significant 320 bytes are the Properties segment, which defines the card capabilities and cannot be modified by the host. The lower 192 bytes are the Modes segment, which defines the configuration the card is working in. These modes can be changed by the host by means of the SWITCH command. The host can use CMD8 (just *e*•MMC supports this command) to get the content of this register.

**RCA register**: The writable 16-bit relative card address register carries the card address that is published by the card during the card identification. This address is used for the addressed host-card communication after the card identification procedure. The host can use CMD3 to ask the card to publish a new relative address (RCA).

**Note**: The default value of the RCA register is 0x0001(*e*•MMC) or 0x0000(SD/SD I/O). The default value is reserved to set all cards into the Stand-by State with CMD7.

**DSR register (Optional)**: The 16-bit driver stage register can be optionally used to improve the bus performance for extended operating conditions (depending on parameters like bus length, transfer rate or number of cards). The CSD register carries the information about the DSR register usage. The default value of the DSR register is 0x404. The host can use CMD4 to get the content of this register.

**SCR register**: Just SD/SD I/O (if has memory port) have this register. In addition to the CSD register, there is another configuration register named SD CARD Configuration Register (SCR), which is only for SD card. SCR provides information on the SD Memory Card's special features that were configured into the given card. The size of SCR register is 64 bits. This register shall be set in the factory by the SD Memory Card manufacturer. The host can use ACMD51 to get the content of this register.

## 36.4.2. Commands

### Commands types

There are four kinds of commands defined to control the Card:

- Broadcast commands (bc), no response
- Broadcast commands with response (bcr) response from all cards simultaneously
- Addressed (point-to-point) commands (ac) no data transfer on DAT
- Addressed (point-to-point) data transfer commands (adtc) data transfer on DAT

### Command format

All commands have a fixed code length of 48 bits, as show in ***Figure 36-9. Command Token Format***, needing a transmission time of 1.92μs (25 MHz) 0.96μs (50 MHz) and 0.92us (52 MHz).

**Figure 36-9. Command Token Format**

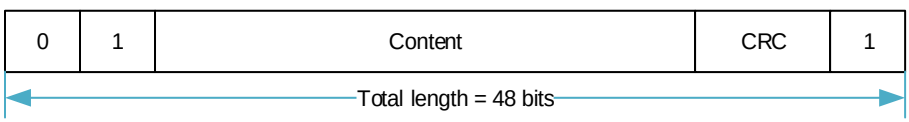


**Table 36-10. Command format**

| Bit position | 47 | 46 | [45:40] | [39:8] | [7:1] | 0 |
|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 32 | 7 | 1 |
| Value | '0' | '1' | x | x | x | '1' |
| Description | start bit | transmission bit | command index | argument | CRC7 | end bit |

A command always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (host = 1). The next 6 bits indicate the index of the command, this value being interpreted as a binary coded number (between 0 and 63). Some commands need an argument (e.g. an address), which is coded by 32 bits. A value denoted by 'x' in the table above indicates this variable is dependent on the command. All commands are protected by a CRC7. Every command code word is terminated by the end bit (always 1).

### Command classes

The command set of the Card system is divided into several classes (See ***Table 36-11. Card command classes (CCCs)***. Each class supports a set of card functionalities. ***Table 36-11. Card command classes (CCCs)*** determines the setting of CCC from the card supported commands.

For SD cards, Class 0, 2, 4, 5 and 8 are mandatory and shall be supported. Class 7 except CMD40 is mandatory for SDHC. The other classes are optional. The supported Card Command Classes (CCC) are coded as a parameter in the card specific data (CSD) register of each card, providing the host with information on how to access the card.

For *e*•MMC cards, Class 0 is mandatory and shall be supported. The other classes are either mandatory only for specific card types or optional. By using different classes, several configurations can be chosen (e.g. a block writable card or a stream readable card). The supported Card Command Classes (CCC) are coded as a parameter in the card specific data (CSD) register of each card, providing the host with information on how to access the card.

**Table 36-11. Card command classes (CCCs)**

| | Card command class(CCC) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Supported command | Class description | basic | Stream read | Block read | Stream write | Block write | erase | write protection | Lock card | application specific | I/O mode | Security Protocols | reserved |
| CMD0 | M | + | | | | | | | | | | | |
| CMD1 | M | + | | | | | | | | | | | |
| CMD2 | M | + | | | | | | | | | | | |
| CMD3 | M | + | | | | | | | | | | | |
| CMD4 | M | + | | | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **CMD5** | M | + | | | | | | | | | + | | |
| **CMD6** | M | + | | | | | | | | | | | |
| **CMD7** | M | + | | | | | | | | | | | |
| **CMD8** | M | + | | | | | | | | | | | |
| **CMD9** | M | + | | | | | | | | | | | |
| **CMD10** | M | + | | | | | | | | | | | |
| **CMD11** | O | + | + | | | | | | | | | | |
| **CMD12** | M | + | | | | | | | | | | | |
| **CMD13** | M | + | | | | | | | | | | | |
| **CMD14** | M | + | | | | | | | | | | | |
| **CMD15** | M | + | | | | | | | | | | | |
| **CMD16** | M | | | + | | + | | | + | | | | |
| **CMD17** | M | | | + | | | | | | | | | |
| **CMD18** | M | | | + | | | | | | | | | |
| **CMD19** | M | + | | | | | | | | | | | |
| **CMD20** | M | | | | + | | | | | | | | |
| **CMD21** | M | | | + | | | | | | | | | |
| **CMD23** | M | | | + | | + | | | | | | | |
| **CMD24** | M | | | | | + | | | | | | | |
| **CMD25** | M | | | | | + | | | | | | | |
| **CMD26** | M | | | | | + | | | | | | | |
| **CMD27** | M | | | | | + | | | | | | | |
| **CMD28** | M | | | | | | | + | | | | | |
| **CMD29** | M | | | | | | | + | | | | | |
| **CMD30** | M | | | | | | | + | | | | | |
| **CMD32** | M | | | | | | + | | | | | | |
| **CMD33** | M | | | | | | + | | | | | | |
| **CMD34** | M | | | | | | | | | | | | + |
| **CMD35** | O | | | | | | + | | | | | | |
| **CMD36** | O | | | | | | + | | | | | | |
| **CMD37** | O | | | | | | | | | | | | + |
| **CMD38** | M | | | | | | + | | | | | | |
| **CMD39** | | | | | | | | | | | + | | |
| **CMD40** | | | | | | | | | | | + | | |
| **CMD42** | | | | | | | | | + | | | | |
| **CMD49** | | | | | | + | | | | | | | |
| **CMD52** | O | | | | | | | | | | + | | |
| **CMD53** | O | | | | | | | | | | + | + | |
| **CMD54** | O | | | | | | | | | | | + | |
| **CMD55** | M | | | | | | | | | + | | | |
| **CMD56** | M | | | | | | | | | + | | | |
| **CMD57** | O | | | | | | | | | | | | + |
| **ACMD6** | M | | | | | | | | | + | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ACMD13** | M | | | | | | | | | + | | | |
| **ACMD22** | M | | | | | | | | | + | | | |
| **ACMD23** | M | | | | | | | | | + | | | |
| **ACMD41** | M | | | | | | | | | + | | | |
| **ACMD42** | M | | | | | | | | | + | | | |
| **ACMD51** | M | | | | | | | | | + | | | |

**Note:** 1. M: mandatory, O: optional.
2. CMD5 is Class 0 command for SD card and Class 9 command in *e*•MMC.
3. In SD cards that support UHS-I, CMD11 is a type 0 command and mandatory, and it is optional in other SD cards; in e•MMC, CMD11 is a type 1 command.
4. CMD14, CMD20, CMD21, CMD26, CMD39, CMD40, CMD49 and CMD54 are only available for *e*•MMC. CMD32, CMD33, CMD52, CMD57 and ACMDx are only available for SD card.
5. All the ACMDx shall be preceded with APP_CMD command (CMD55).
6. CMD5, CMD8 , CMD11 and CMD53 has different meaning for *e*•MMC and SD memory.
7. Class 1 and Class 3 are obsolete.

### Detailed command description

The following tables describe in detail all bus commands. The responses R1-R7 are defined in

***Responses***. The registers CID, CSD and DSR are described in ***Card registers***. The card shall ignore stuff bits and reserved bits in an argument.

**Table 36-12. Basic commands (class 0)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD0 | bc | [31:0] 00000000 | - | GO_IDLE_STATE | Resets all cards to idle state |
| | bc | [31:0] F0F0F0F0 | - | GO_PRE_IDLE_STATE | Resets all cards to pre-idle state |
| | - | [31:0] FFFFFFFA | - | BOOT_INITIATION | Initiate alternative boot operation |
| CMD1 | bc | [31:0] OCR without busy | R3 | SEND_OP_COND | Asks the card, in idle state, to send its Operating Conditions Register contents in the response on the CMD line. |
| CMD2 | bcr | [31:0] stuff bits | R2 | ALL_SEND_CID | Asks any card to send the CID numbers on the CMD line (any card that is connected to the host will respond) |
| CMD3 | bcr | [31:0] stuff bits | R6 | SEND_RELATIVE_ADDR | Ask the card to publish a new relative address (RCA) |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD4 | bc | [31:16] DSR<br>[15:0] stuff bits | - | SET_DSR | Programs the DSR of all cards |
| CMD5 | bcr | [31:25]reserved bits<br>[24]S18R<br>[23:0] I/O OCR | R4 | IO_SEND_OP_COND | Only for I/O cards. It is similar to the operation of ACMD41 for SD memory cards, used to inquire about the voltage range needed by the I/O card. |
| CMD5 | ac | [31:16] RCA<br>[15] Sleep/Awake<br>[14:0] stuff bits | R1b | SLEEP_AWAKE | Only for *e*•MMC. Toggles the Device between Sleep state and Standby state. |
| CMD6 | adtc | [31] Mode<br>0:Check function<br>1:Switch function<br>[30:24] reserved<br>[23:20] reserved for function group 6 (0h or Fh)<br>[19:16] reserved for function group 5 (0h or Fh)<br>[15:12] reserved for function group 4 (0h or Fh)<br>[11:8] reserved for function group 3 (0h or Fh)<br>[7:4] function group 2 for command system<br>[3:0] function group 1 for access mode | R1 | SWITCH_FUNC | Only for SD memory and SD I/O. Checks switchable function (mode 0) and switch card function (mode 1). |
| CMD6 | ac | [31:26] Set to 0<br>[25:24] Access<br>[23:16] Index<br>[15:8] Value<br>[7:3] Set to 0<br>[2:0] Cmd Set | R1b | SWITCH | Only for *e*•MMC. Switches the mode of operation of the selected card or modifies the EXT_CSD registers. |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD7 | ac | [31:16] RCA<br>[15:0] stuff bits | R1b | SELECT/DESELECT_CARD | Command toggles a card between the stand-by and transfer states or between the programming and disconnects states. In both cases the card is selected by its own relative address and gets deselected by any other address; address 0 deselects the card. |
| CMD8 | bcr | [31:12]reserved bits<br>[11:8]supply voltage(VHS)<br>[7:0]check pattern | R7 | SEND_IF_COND | Sends SD Memory Card interface condition, which includes host supply voltage information and asks the card whether card supports voltage. Reserved bits shall be set to '0'. |
| CMD8 | adtc | [31:0] stuff bits | R1 | SEND_EXT_CSD | For *e*•MMC only. The card sends its EXT_CSD register as a block of data. |
| CMD9 | ac | [31:16] RCA<br>[15:0] stuff bits | R2 | SEND_CSD | Addressed card sends its card-specific data (CSD) on the CMD line. |
| CMD10 | ac | [31:16] RCA<br>[15:0] stuff bits | R2 | SEND_CID | Addressed card sends its card identification (CID) on CMD the line. |
| CMD11 | ac | [31:0] 00000000 | R1 | VOLTAGE_SWITCH | Switch to 1.8V bus signaling level. |
| CMD12 | ac | [31:0] stuff bits | R1b | STOP TRANSMISSION | Forces the card to stop transmission |
| CMD13 | ac | [31:16] RCA<br>[15:0] stuff bits | R1 | SEND_STATUS | Addressed card sends its status register. |
| CMD14 | adtc | [31:0] stuff bits | R1 | BUSTEST_R | A host reads the reversed bus testing data pattern from a card. |
| CMD15 | ac | [31:16] RCA<br>[15:0] reserved bits | - | GO_INACTIVE_STATE | Sends an addressed card into the Inactive State. This command is used when the host explicitly wants to deactivate a card. |
| CMD19 | adtc | [31:0] stuff bits | R1 | BUSTEST_W | A host sends the bus test data pattern to a card. |

**Table 36-13. Block-Oriented read commands (class 2)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD16 | ac | [31:0] block length | R1 | SET_BLOCKLEN | In the case of a Standard Capacity SD and *e*•MMC, this command sets the block length |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| | | | | | (in bytes) for all following block commands (read, write, lock). Default is 512 Bytes. Set length is valid for memory access commands only if partial block read operation are allowed in CSD.<br>In the case of a High Capacity SD Memory Card, block length set by CMD16 command does not affect the memory read and write commands. Always 512 Bytes fixed block length is used. In both cases, if block length is set larger than 512Bytes, the card sets the BLOCK_LEN_ERROR bit. |
| CMD17 | adtc | [31:0] data address | R1 | READ_SINGLE_BLOCK | In the case of a Standard Capacity SD and *e*•MMC, this command reads a block of the size selected by the SET_BLOCKLEN command.<br>In the case of a High Capacity Card, block length is fixed 512 Bytes regardless of the SET_BLOCKLEN command. |
| CMD18 | adtc | [31:0] data address | R1 | READ_MULTIPLE_BLOCK | Continuously transfers data blocks from card to host until interrupted by a STOP_TRANSMISSION command. Block length is specified the same as READ_SINGLE_BLOCK command. |
| CMD21 | adtc | [31:0] stuff bits | R1 | SEND_TUNNING_BLOCK | 128 clocks of tuning pattern (64byte in 4 bit mode or 128byte in 8 bit mode) is sent for HS200 optimal sampling point detection. |
| **Note:** The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register | | | | | |

**Table 36-14. Stream read commands (class 1) and stream write commands (class 3)**

| Cmd | type | argument | Response | Abbreviation | Description |
|---|---|---|---|---|---|

| index | | | format | | |
|---|---|---|---|---|---|
| CMD11 | adtc | [31:0] data address | R1 | READ_DAT_UNTIL_STOP | Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows. |
| CMD20 | adtc | [31:0] data address | R1 | WRITE_DAT_UNTIL_STOP | Writes data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows. |
| **Note:** The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register | | | | | |

**Table 36-15. Block-Oriented write commands (class 4)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD16 | ac | [31:0] block length | R1 | SET_BLOCKLEN | See description in ***Table 36-13. Block-Oriented read commands (class 2)***. |
| CMD23 | ac | [31:16] set to 0 [15:0] number of blocks | R1 | SET_BLOCK_COUNT | Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command. If the argument is all 0s, the subsequent read/write operation will be open-ended. |
| CMD24 | adtc | [31:0] data address | R1 | WRITE_BLOCK | In the case of a Standard Capacity SD, this command writes a block of the size selected by the SET_BLOCKLEN command. In the case of a SDHC, block length is fixed 512 Bytes regardless of the SET_BLOCKLEN command. |
| CMD25 | adtc | [31:0] data address | R1 | WRITE_MULTIPLE _BLOCK | Continuously writes blocks of data until a STOP_TRANSMISSION follows. Block length is specified the same as WRITE_BLOCK command. |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD26 | adtc | [31:0] stuff bits | R1 | PROGRAM_CID | Programming of the card identification register. This command shall be issued only once. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer. |
| CMD27 | adtc | [31:0] stuff bits | R1 | PROGRAM_CSD | Programming of the programmable bits of the CSD. |
| CMD49 | adtc | [31:0] stuff bits | R1 | SET_TIME | Sets the real time clock according to the RTC information in the 512B data block. |
| **Note:** 1. The data transferred shall not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD. In the case that write partial blocks is not supported, then the block length=default block length (given in CSD).<br>2. Data address is in byte units in a Standard Capacity SD Memory Card and in block (512 Byte) units in a High Capacity SD Memory Card. | | | | | |

**Table 36-16. Erase commands (class 5)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD32 | ac | [31:0] data address | R1 | ERASE_WR_BLK_START | Sets the address of the first write block to be erased.(SD) |
| CMD33 | ac | [31:0] data address | R1 | ERASE_WR_BLK_END | Sets the address of the last write block of the continuous range to be erased.(SD) |
| CMD35 | ac | [31:0]data address | R1 | ERASE_GROUP_START | Sets the address of the first erase group within a range to be selected for erase.( *e*•MMC) |
| CMD36 | ac | [31:0]data address | R1 | ERASE_GROUP_END | Sets the address of the last erase group within a continuous range to be selected for erase. ( *e*•MMC) |
| CMD38 | ac | [31:0] stuff bits | R1b | ERASE | Erases all previously selected write blocks. |
| **Note:** 1.CMD34 and CMD37 are reserved in order to maintain backwards compatibility with older versions of the *e*•MMC.<br>2. Data address is in byte units in a Standard Capacity SD Memory Card and in block (512 Byte) units in a High Capacity SD Memory Card. | | | | | |

**Table 36-17. Block oriented write protection commands (class 6)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD28 | ac | [31:0] data address | R1b | SET_WRITE_PROT | If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).<br>A High Capacity SD Memory Card does not support this command. |
| CMD29 | ac | [31:0] data address | R1b | CLR_WRITE_PROT | If the card provides write protection features, this command clears the write protection bit of the addressed group. |
| CMD30 | adtc | [31:0] write protect data address | R1 | SEND_WRITE_PROT | If the card provides write protection features, this command asks the card to send the status of the write protection bits. |
| **Note:** 1. High Capacity SD Memory Card does not support these three commands. | | | | | |

**Table 36-18. Lock card (class 7)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD16 | ac | [31:0] block length | R1 | SET_BLOCK_LEN | See description in ***Table 36-13. Block-Oriented read commands (class 2)***. |
| CMD42 | adtc | [31:0] Reserved bits (Set all 0) | R1 | LOCK_UNLOCK | Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.<br>Reserved bits in the argument and in Lock Card Data Structure shall be set to 0. |

**Table 36-19. Application-specific commands (class 8)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| ACMD41 | bcr | [31]reserved bit<br>[30]HCS<br>[29:24]reserved | R3 | SD_SEND_OP_COND | Sends host capacity support information (HCS) and asks the accessed card to send its |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| | | bits [23:0]$V_{DD}$ Voltage Window(OCR[23:0]) | | | operating condition register(OCR) content in the response. HCS is effective when card receives SEND_IF_COND command. CCS bit is assigned to OCR[30]. |
| ACMD42 | ac | [31:1] stuff bits [0]set_cd | R1 | SET_CLR_CARD_DETECT | Connect[1]/Disconnect[0] the 50K pull-up resistor on CD/DAT3 (pin 1) of the card. |
| ACMD51 | adtc | [31:0] stuff bits | R1 | SEND_SCR | Reads the SD Configuration Register (SCR). |
| CMD55 | ac | [31:16] RCA [15:0] stuff bits | R1 | APP_CMD | Indicates to the card that the next command is an application specific command rather than a standard command. |
| CMD56 | adtc | [31:1] stuff bits. [0] RD/WR | R1 | GEN_CMD | Used either to transfer a data block to the card or to get a data block from the card for general purpose/application specific command. The host sets RD/WR=1 for reading data from the card and sets to 0 for writing data to the card. |
| **Note:** 1. ACMDx is Application-specific Commands for SD memory. | | | | | |

**Table 36-20. I/O mode commands (class 9)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD39 | ac | [31:16] RCA [15] register write flag [14:8] register address [7:0] register data | R4 | FAST_IO | Used to write and read 8 bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register if the write flag is cleared to 0. This command accesses application dependent registers which are |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| | | | | | not defined in the *e*•MMC standard. |
| CMD40 | bcr | [31:0] stuff bits | R5 | GO_IRQ_STATE | Sets the system into interrupt mode |
| CMD52 | adtc | [31] R/W Flag<br>[30:28] Function Number<br>[27] RAW Flag<br>[26] stuff bits<br>[25:9] Register Address<br>[8] stuff bits<br>[7:0] Write Data/Stuff bits | R5 | IO_RW_DIRECT | The IO_RW_DIRECT is the simplest means to access a single register within the total 128K of register space in any I/O function, including the common I/O area (CIA). This command reads or writes 1 byte using only 1 command/response pair. A common use is to initialize registers or monitor status values for I/O functions. This command is the fastest means to read or write single I/O registers, as it requires only a single command/response pair. |
| CMD53 | adtc | [31] R/W Flag<br>[30:28] Function Number<br>[27] Block Mode<br>[26] OP code<br>[25:9] Register Address<br>[8:0] Byte/Block Count | | IO_RW_EXTENDED | This command allows the reading or writing of a large number of I/O registers with a single command. |
| **Note:** 1.CMD39, CMD40 are only for *e*•MMC.<br>2. CMD52, CMD53 are only for SD I/O card. | | | | | |

**Table 36-21. Switch function commands (class 10)**

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| CMD53 | adtc | [31:16] Security Protocol Specific<br>[15:8] Security Protocol<br>[7:0] reserved | R1 | PROTOCOL_RD | Only for SD memory and SD I/O. Continuously transfers data blocks from device to host. Number of data blocks shall be defined by a preceding CMD23. |

| Cmd index | type | argument | Response format | Abbreviation | Description |
|---|---|---|---|---|---|
| | | | | | Data Transfer may be interrupted by a STOP_TRANSMISSION command, This command is not supported if sent as a packed command. Block size is always 512bytes |
| CMD54 | adtc | [31:16] Security Protocol Specific<br>[15:8] Security Protocol<br>[7:0] reserved | R1 | PROTOCOL_WR | Only for SD memory and SD I/O. Continuously transfers data blocks from host to device. Number of data blocks shall be defined by a preceding CMD23. Data Transfer may be interrupted by a STOP_TRANSMISSION command, This command is not supported if sent as a packed command. Block size is always 512bytes. |

### 36.4.3. Responses

All responses are sent on the CMD line. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type.

#### Responses types

There are 7 types of responses show as follows.

- **R1 / R1b :** normal response command.
- **R2 :** CID, CSD register.
- **R3 :** OCR register.
- **R4 :** Fast I/O.
- **R5 :** Interrupt request.
- **R6 :** Published RCA response.

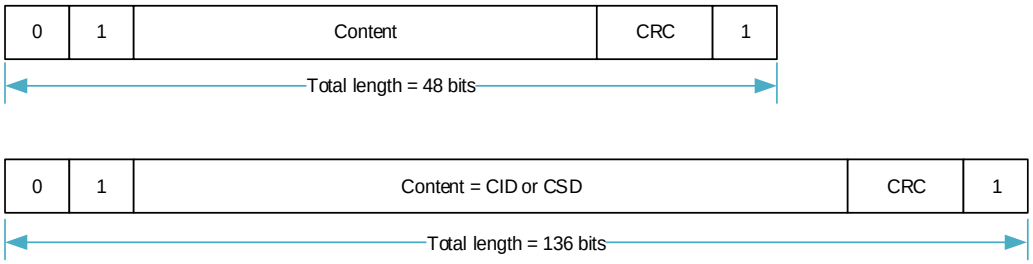
- **R7 :** Card interface condition.

The SD Memory Card support five types of them, R1 / R1b, R2, R3, R6, R7. And the SD I/O Card and *e*•MMC supports additional response types named R4 and R5, but they are not exactly the same for SD I/O Card and *e*•MMC.

### Responses format

Responses have two formats, as show in ***Figure 36-10. Response Token Format***, all responses are sent on the CMD line. The code length depends on the response type. Except R2 is 136 bits length, others are all 48 bits length.

**Figure 36-10. Response Token Format**




A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value ‘x’ in the tables below indicates a variable entry. All responses except for the type R3 are protected by a CRC. Every command code word is terminated by the end bit (always 1).

### R1 (normal response command)

Code length is 48 bits. The bits 45:40 indicate the index of the command to be responded to, this value being interpreted as a binary coded number (between 0 and 63). The status of the card is coded in 32 bits. Note that if a data transfer to the card is involved, then a busy signal may appear on the data line after the transmission of each block of data. The host shall check for busy after data block transmission. The card status is described in ***Two status fields of the card***.

**Table 36-22. Response R1**

| **Bit position** | 47 | 46 | [45:40] | [39:8] | [7:1] | 0 |
|---|---|---|---|---|---|---|
| **Width** | 1 | 1 | 6 | 32 | 7 | 1 |
| **Value** | ‘0’ | ‘0’ | x | x | x | ‘1’ |
| **description** | start bit | transmission bit | command index | card status | CRC7 | end bit |

### R1b

R1b is identical to R1 with an optional busy signal transmitted on the data line DAT0. The card may become busy after receiving these commands based on its state prior to the command reception. The Host shall check for busy at the response.

### R2 (CID, CSD register)

Code length is 136 bits. The contents of the CID register are sent as a response to the commands CMD2 and CMD10. The contents of the CSD register are sent as a response to CMD9. Only the bits [127..1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response.

**Table 36-23. Response R2**

| Bit position | 135 | 134 | [133:128] | [127:1] | 0 |
|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 127 | 1 |
| Value | '0' | '0' | '111111' | x | '1' |
| description | start bit | transmission bit | reserved | CID or CSD register and internal CRC7 | end bit |

### R3 (OCR register)

Code length is 48 bits. The contents of the OCR register are sent as a response to ACMD41 (SD memory), CMD1 (*e*•MMC). The response of different cards may have a little different.

**Table 36-24. Response R3**

| Bit position | 47 | 46 | [45:40] | [39:8] | [7:1] | 0 |
|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 32 | 7 | 1 |
| Value | '0' | '0' | '111111' | x | '1111111' | '1' |
| description | start bit | transmission bit | reserved | OCR register | reserved | end bit |

### R4 (Fast I/O)

For *e*•MMC only. Code length 48 is bits. The argument field contains the RCA of the addressed card, the register address to be read out or written to, and its contents. The status bit in the argument is set if the operation was successful.

**Table 36-25. Response R4 for e•MMC**

| Bit position | 47 | 46 | [45:40] | [39:8] Argument field | | | | [7:1] | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 16 | 1 | 7 | 8 | 7 | 1 |
| Value | '0' | '0' | '100111' | x | x | x | x | x | '1' |
| description | start bit | transmission bit | CMD39 | RCA [31:16] | status [15] | register address [14:8] | read register contents [7:0] | CRC7 | end bit |

### R4b

For SD I/O only. Code length is 48 bits. The SDIO card receive the CMD5 will respond with a unique SD I/O response R4.

**Table 36-26. Response R4 for SD I/O**

| Bit position | 47 | 46 | [45:40] | 39 | [38:36] | 35 | [34:32] | 31 | [30:8] | [7:1] | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 1 | 3 | 1 | 3 | 1 | 23 | 7 | 1 |
| Value | '0' | '0' | '1111 11' | x | x | x | '000' | x | x | '1111 111' | 1 |
| description | start bit | transmission bit | Reserved | C | Number of I/O functions | Memory Present | Stuff Bits | S18A | I/O OCR | Reserved | end bit |

### R5 (Interrupt request)

For *e*•MMC only. Code length is 48 bits. If the response is generated by the host, the RCA field in the argument will be 0x0.

**Table 36-27. Response R5 for *e*•MMC**

| Bit position | 47 | 46 | [45:40] | [39:8] Argument field | | [7:1] | 0 |
|---|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 16 | 16 | 7 | 1 |
| Value | '0' | '0' | '101000' | x | x | x | '1' |
| description | start bit | transmission bit | CMD40 | RCA [31:16] of winning card or of the host | [15:0] Not defined. May be used for IRQ data | CRC7 | end bit |

### R5b

For SD I/O only. The SDIO card's response to CMD52 and CMD53 is R5. If the communication between the card and host is in the 1-bit or 4-bit SD mode, the response shall be in a 48-bit response (R5).

**Table 36-28. Response R5 for SD I/O**

| Bit position | 47 | 46 | [45:40] | [39:24] | [23:16] | [15:8] | [7:1] | 0 |
|---|---|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 16 | 8 | 8 | 7 | 1 |
| Value | '0' | '0' | '11010x' | '0' | x | x | x | '1' |
| description | start bit | transmission bit | CMD52/53 | Stuff Bits | Response Flags | Read or Write Data | CRC7 | end bit |

### R6 (Published RCA response)

Code length is 48 bit. The bits [45:40] indicate the index of the command to be responded to (CMD3). The 16 MSB bits of the argument field are used for the Published RCA number.

**Table 36-29. Response R6**

| Bit position | 47 | 46 | [45:40] | [39:8] Argument field | | [7:1] | 0 |
|---|---|---|---|---|---|---|---|
| Width | 1 | 1 | 6 | 16 | 16 | 7 | 1 |
| Value | '0' | '0' | '000011' | x | x | x | '1' |
| description | start | transmission | CMD3 | New published RCA | card status | CRC7 | end |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | bit | bit | | of the card | bits:23,22,19,12:0 | | bit |

### R7 (Card interface condition)

For SD memory only. Code length is 48 bits. The card support voltage information is sent by the response of CMD8. Bits 19-16 indicate the voltage range that the card supports. The card that accepted the supplied voltage returns R7 response. In the response, the card echoes back both the voltage range and check pattern set in the argument.

**Table 36-30. Response R7**

| **Bit position** | 47 | 46 | [45:40] | [39:20] | [19:16] | [15:8] | [7:1] | 0 |
|---|---|---|---|---|---|---|---|---|
| **Width** | 1 | 1 | 6 | 20 | 4 | 8 | 7 | 1 |
| **Value** | '0' | '0' | '001000' | '00000h' | x | x | x | '1' |
| **description** | start bit | transmission bit | CMD8 | Reserved bits | Voltage accepted | echo-back of check pattern | CRC7 | end bit |

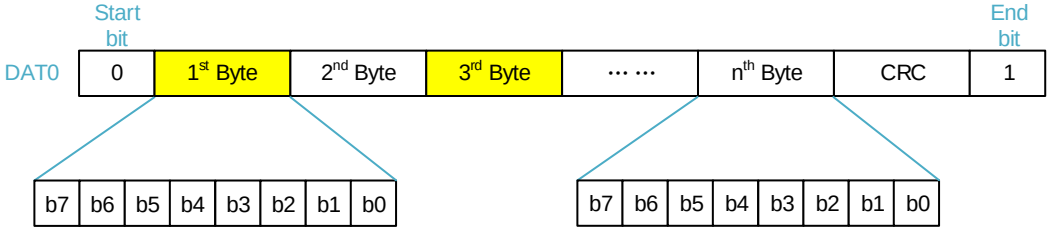
## 36.4.4. Data packets format

There are 5 data bus mode, 1-bit, 4-bit SDR/DDR, 8-bit SDR/DDR. 1-bit mode is mandatory, 4-bit and 8-bit mode is optional.

### 1-bit data packet format

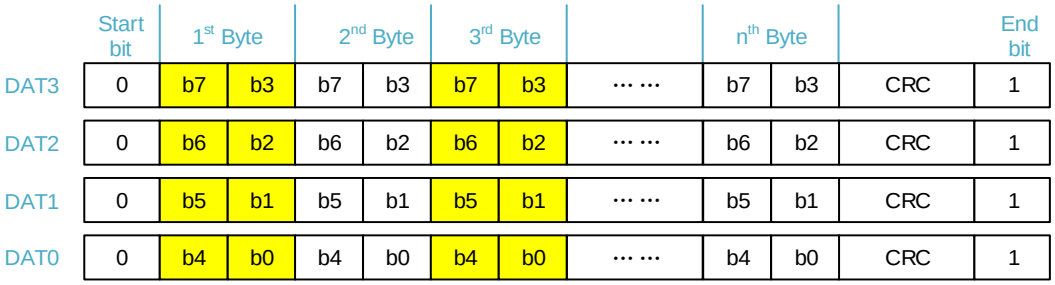
After card reset and initialize, only DAT0 pin is used to transfer data. And other pin can be used freely. ***Figure 36-11. 1-bit data bus width***, ***Figure 36-12. 4-bit data bus width*** and ***Figure 36-13. 8-bit data bus width*** show the data packet format when data bus wide is 1-bit, 4-bit and 8-bit.

**Figure 36-11. 1-bit data bus width**



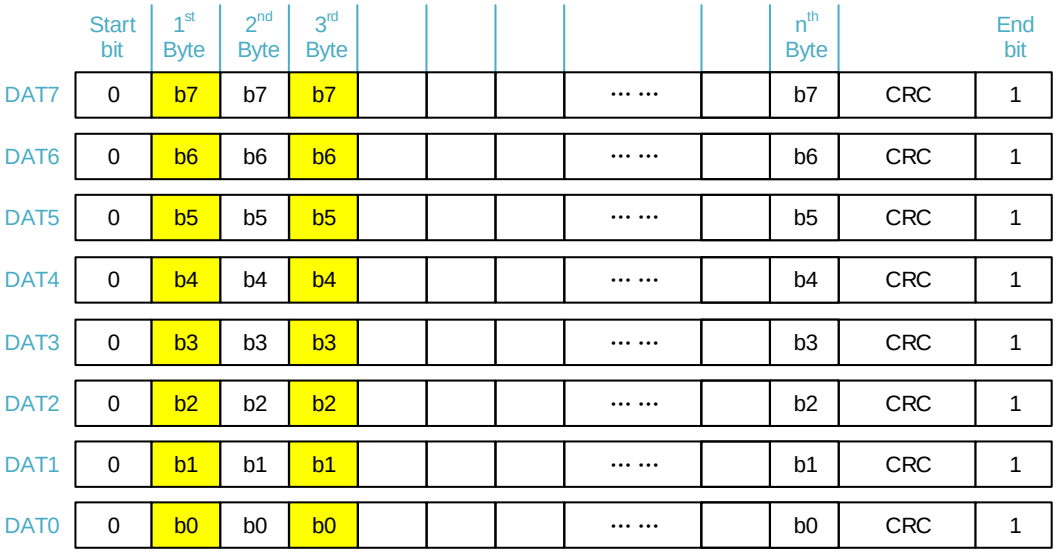

### 4-bit data packet format

**Figure 36-12. 4-bit data bus width**



### 8-bit data packet format

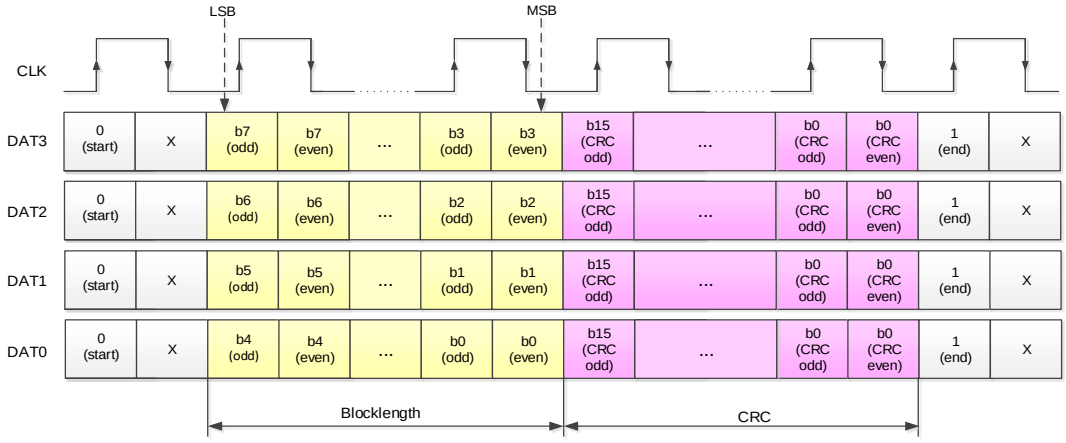
**Figure 36-13. 8-bit data bus width**




### 4-bit bus data packet format for DDR (DAT3-DAT0 used)

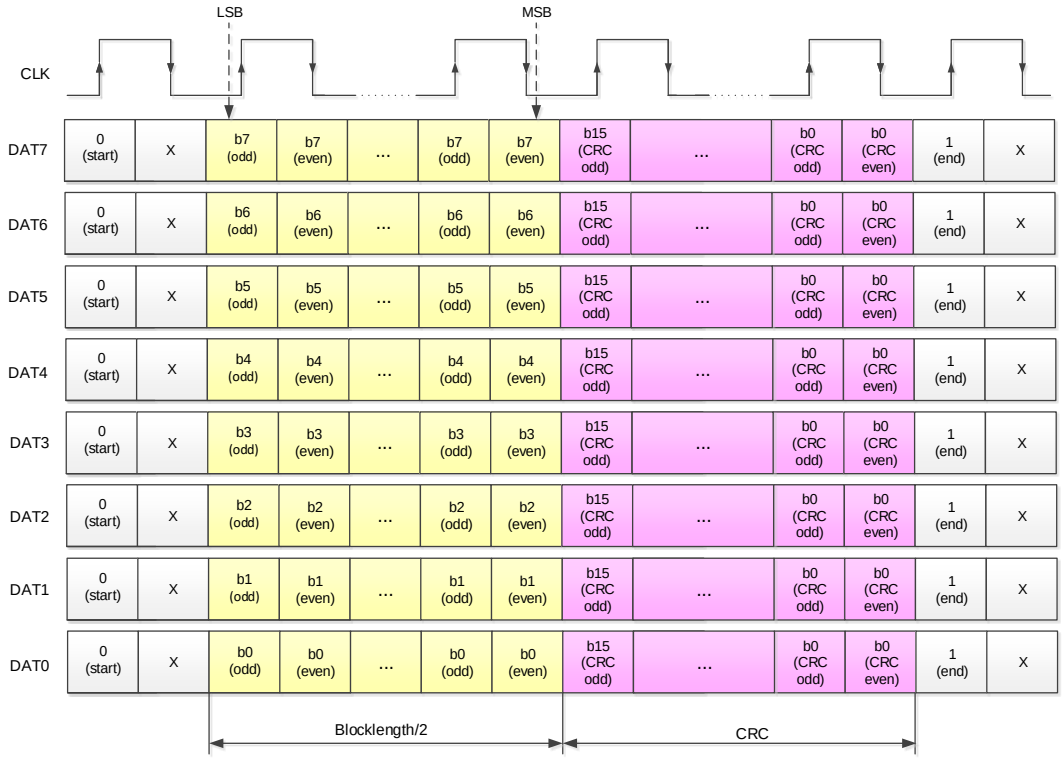
For each data lines, data can be transferred at the rate of one bit (single data rate) or two bits (double data rate) per clock cycle. DDR (double data rate) signaling, data is sampled on both SDIO_CLK clock edges. ***Figure 36-14. 4-bit data bus DDR*** and ***Figure 36-15. 8-bit data bus DDR*** show the data packet format when data bus wide is 4-bit DDR and 8-bit DDR.

**Figure 36-14. 4-bit data bus DDR**



**8-bit bus data packet format for DDR (DAT7-DAT0 used)**

**Figure 36-15. 8-bit data bus DDR**




**Note:** For DDR data bus: 1.That bytes data are not interleaved but CRC are interleaved. 2. Start bits are valid on the rising and falling edge. 3. End bits are only valid on the rising edge (“x” undefined).

## 36.4.5. Two status fields of the card

The SD Memory supports two status fields and others just support the first one:

Card Status: Error and state information of a executed command, indicated in the response

SD Status: Extended status field of 512 bits that supports special features of the SD Memory Card and future Application-Specific features.

### Card status

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card’s status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

The type and clear condition fields in the table are abbreviated as follows:

**Type**

•E: Error bit. Send an error condition to the host. These bits are cleared as soon as the

response (reporting the error) is sent out.

•S: Status bit. These bits serve as information fields only, and do not alter the execution of the command being responded to. These bits are persistent, they are set and cleared in accordance with the card status.

•R: Exceptions are detected by the card during the command interpretation and validation phase (Response Mode).

•X: Exceptions are detected by the card during command execution phase (Execution Mode).

**Clear condition**

•A: According to current state of the card.

•B: Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).

•C: Cleared by read

**Table 36-31. Card status**

| Bits | Identifier | Type | Value | Description | Clear Condition |
|---|---|---|---|---|---|
| 31 | OUT_OF_RANGE | ERX | ’0’= no error<br>’1’= error | The command’s argument was out of the allowed range for this card. | C |
| 30 | ADDRESS_ERROR | ERX | ’0’= no error<br>’1’= error | A misaligned address which did not match the block length was used in the command. | C |
| 29 | BLOCK_LEN_ERROR | ERX | ’0’= no error<br>’1’= error | The transferred block length is not allowed for this card, or the number of transferred bytes does not match the block length. | C |
| 28 | ERASE_SEQ_ERROR | ER | ’0’= no error<br>’1’= error | An error in the sequence of erase commands occurred. | C |
| 27 | ERASE_PARAM | ERX | ’0’= no error<br>’1’= error | An invalid selection of write-blocks for erase occurred. | C |
| 26 | WP_VIOLATION | ERX | ’0’= not protected<br>’1’= protected | Set when the host attempts to write to a protected block or to the temporary or permanent write protected card. | C |
| 25 | CARD_IS_LOCKED | SX | ‘0’ = card unlocked<br>‘1’ = card locked | When set, signals that the card is locked by the host | A |
| 24 | LOCK_UNLOCK_FAILED | ERX | ’0’= no error<br>’1’= error | Set when a sequence or password error has been detected in lock/unlock card command. | C |

| Bits | Identifier | Type | Value | Description | Clear Condition |
|---|---|---|---|---|---|
| 23 | COM_CRC_ERROR | ER | ’0’= no error<br>’1’= error | The CRC check of the previous command failed. | B |
| 22 | ILLEGAL_COMMAND | ER | ’0’= no error<br>’1’= error | Command not legal for the card state. | B |
| 21 | CARD_ECC_FAILED | ERX | ’0’= success<br>’1’= failure | Card internal ECC was applied but failed to correct the data. | C |
| 20 | CC_ERROR | ERX | ’0’= no error<br>’1’= error | Internal card controller error. | C |
| 19 | ERROR | ERX | ’0’= no error<br>’1’= error | A general or an unknown error occurred during the operation. | C |
| 18 | UNDERRUN | ERX | ’0’= no error<br>’1’= error | Only for *e*•MMC. The card could not sustain data transfer in stream read mode. | C |
| 17 | OVERRUN | ERX | ’0’= no error<br>’1’= error | Only for *e*•MMC. The card could not sustain data programming in stream write mode. | C |
| 16 | CID/ CSD_OVERWRITE | ERX | ’0’= no error<br>’1’= error | Can be either one of the following errors:<br>- The read only section of the CSD does not match the card content.<br>- An attempt to reverse the copy (set as original) or permanent WP(unprotected) bits was made. | C |
| 15 | WP_ERASE_SKIP | ERX | ’0’= not protected<br>’1’= protected | Set when only partial address space was erased due to existing write protected blocks or the temporary or permanent write protected card was erased. | C |
| 14 | CARD_ECC_DISABLED | SX | ’0’= enabled<br>’1’= disabled | The command has been executed without using the internal ECC. | A |
| 13 | ERASE_RESET | SR | ’0’= cleared<br>’1’= set | An erase sequence was cleared before executing because an out of erase sequence command was received. | C |
| [12:9] | CURRENT_STATE | SX | 0 = idle<br>1 = ready | The state of the card when receiving the command. If the | B |

| Bits | Identifier | Type | Value | Description | Clear Condition |
|---|---|---|---|---|---|
| | | | 2 = identification<br>3 = stand by<br>4 = transfer<br>5 = send data<br>6 = receive data<br>7 = programming<br>8 = disconnect<br>9 = Bus-Test<br>10 = sleep<br>11-14 = reserved<br>15 = reserved for I/O mode | command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as a binary coded number between 0 and 15.<br>Sleep state for for *e*•MMC only. | |
| 8 | READY_FOR_DATA | SX | '0'= not ready<br>'1'= ready | Corresponds to buffer empty signaling on the bus. | A |
| 7 | SWITCH_ERROR | EX | '0'= no error<br>'1'= switch error | If set, the card don't switch to the expected mode as requested by the SWITCH command. | B |
| 6 | Reserved | | | | |
| 5 | APP_CMD | SR | '0'= enabled<br>'1'= disabled | The card will expect ACMD, or an indication that the command has been interpreted as ACMD. | C |
| 4 | Reserved | | | | |
| 3 | AKE_SEQ_ERROR | ER | '0'= no error<br>'1'= error | Only for SD memory. Error in the sequence of the authentication process. | C |
| 2 | Reserved for application specific commands. | | | | |
| [1:0] | Reserved for manufacturer test mode. | | | | |

**Note:** 18, 17, 7 bits are only for *e*•MMC. 14, 3 bits are only for SD memory.

### SD status register

The SD Status contains status bits that are related to the SD Memory Card proprietary features and may be used for future application-specific usage. The size of the SD Status is one data block of 512 bits. The content of this register is transmitted to the Host over the DAT bus along with a 16-bit CRC. The SD Status is sent to the host over the DAT bus as a response to ACMD13 (CMD55 followed with CMD13). ACMD13 can be sent to a card only in 'transfer state' (card is selected). The SD Status structure is described below.

The same abbreviation for 'type' and 'clear condition' were used as for the Card Status above.

**Table 36-32. SD status**

| Bits | Identifier | Type | Value | Description | Clear Condition |
|---|---|---|---|---|---|
| [511:510] | DAT_BUS_WIDTH | SR | '00'= 1 (default)<br>'01'= reserved<br>'10'= 4 bit width<br>'11'= reserved | Shows the currently defined data bus width that was defined by SET_BUS_WIDTH command | A |
| 509 | SECURED_MODE | SR | '0'= Not in the mode<br>'1'= In Secured Mode | Card is in Secured Mode of operation (refer to the "SD Security Specification"). | A |
| [508:496] | reserved | | | | |
| [495:480] | SD_CARD_TYPE | SR | The following cards are currently defined:<br>'0000'= Regular SD RD/WR Card.<br>'0001'= SD ROM Card<br>'0002'= OTP | In the future, the 8 LSBs will be used to define different variations of an SD Memory Card (Each bit will define different SD Types). The 8 MSBs will be used to define SD Cards that do not comply with current SD Physical Layer Specification. | A |
| [479:448] | SIZE_OF_PROTECTED_AREA | SR | Size of protected area | (See below) | A |
| [447:440] | SPEED_CLASS | SR | Speed class of the card | (See below) | A |
| [439:432] | PERFORMANCE_MOVE | SR | Performance of move indicated by 1 [MB/s] step. | (See below) | A |
| [431:428] | AU_SIZE | SR | Size of AU | (See below) | A |
| [427:424] | reserved | | | | |
| [423:408] | ERASE_SIZE | SR | Number of AUs to be erased at a time | (See below) | A |
| [407:402] | ERASE_TIMEOUT | SR | Timeout value for erasing areas specified by UNIT_OF_ERASE_AU | (See below) | A |
| [401:400] | ERASE_OFFSET | SR | Fixed offset value added to erase time. | (See below) | A |

| Bits | Identifier | Type | Value | Description | Clear Condition |
|---|---|---|---|---|---|
| [399:312] | reserved | | | | |
| [311:0] | reserved for manufacturer | | | | |

**SIZE_OF_PROTECTED_AREA**

Setting this field differs between SDSC and SDHC/SDXC.

In case of SDSC Card, the capacity of protected area is calculated as follows:

Protected Area = SIZE_OF_PROTECTED_AREA* MULT * BLOCK_LEN.

SIZE_OF_PROTECTED_AREA is specified by the unit in MULT*BLOCK_LEN.

In case of SDHC and SDXC Cards, the capacity of protected area is calculated as follows:

Protected Area = SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA is specified by the unit in byte.

**SPEED_CLASS**

This 8-bit field indicates the Speed Class.

00h: Class 0

01h: Class 2

02h: Class 4

03h: Class 6

04h: Class 10

05h–FFh: Reserved

**PERFORMANCE_MOVE**

This 8-bit field indicates Pm and the value can be set by 1 [MB/sec] step. If the card does not move useing RUs, Pm should be considered as infinity. Setting to FFh means infinity. The minimum value of Pm is defined in ***Table 36-33. Performance move field***.

**Table 36-33. Performance move field**

| PERFORMANCE_MOVE | Value Definition |
|---|---|
| 00h | Sequential Write |
| 01h | 1 [MB/sec] |
| 02h | 2 [MB/sec] |
| ..... | ..... |
| FEh | 254 [MB/sec] |
| FFh | Infinity |

**AU_SIZE**

This 4-bit field indicates AU Size and the value can be selected from 16 KB.

**Table 36-34. AU_SIZE field**

| AU_SIZE | Value Definition |
|---|---|
| 0h | Not Defined |
| 1h | 16 KB |
| 2h | 32 KB |
| 3h | 64 KB |
| 4h | 128 KB |
| 5h | 256 KB |
| 6h | 512 KB |
| 7h | 1 MB |
| 8h | 2 MB |
| 9h | 4 MB |
| Ah | 8 MB |
| Bh | 12 MB |
| Ch | 16 MB |
| Dh | 24 MB |
| Eh | 32 MB |
| Fh | 64 MB |

The maximum AU size, depends on the card capacity, is defined in ***Table 36-34. AU_SIZE field***. The card can set any AU size specified in ***Table 36-35. Maximum AU size*** that is less than or equal to the maximum AU size. The card should set smaller AU size as possible.

**Table 36-35. Maximum AU size**

| Card Capacity | up to 64MB | up to 256MB | up to 512MB | up to 32GB | up to 2TB |
|---|---|---|---|---|---|
| Maximum AU Size | 512 KB | 1 MB | 2 MB | 4 MB1 | 64MB |

**ERASE_SIZE**

This 16-bit field indicates $N_{ERASE}$. When $N_{ERASE}$ of AUs are erased, the timeout value is specified by ERASE_TIMEOUT (Refer to ERASE_TIMEOUT). The host should determine proper number of AUs to be erased in one operation so that the host can indicate progress of erase operation. If this field is set to 0, the erase timeout calculation is not supported.

**Table 36-36. Erase size field**

| ERASE_SIZE | Value Definition |
|---|---|
| 0000h | Erase Time-out Calculation is not supported. |
| 0001h | 1 AU |
| 0002h | 2 AU |
| 0003h | 3 AU |
| ..... | ..... |
| FFFFh | 65535 AU |

**ERASE_TIMEOUT**

This 6-bit field indicates the $T_{ERASE}$ and the value indicates erase timeout from offset when multiple AUs are erased as specified by ERASE_SIZE. The range of ERASE_TIMEOUT can be defined as up to 63 seconds and the card manufacturer can choose any combination of ERASE_SIZE and ERASE_TIMEOUT depending on the implementation. Once ERASE_TIMEOUT is determined, it determines the ERASE_SIZE. The host can determine timeout for any number of AU erase by the equation below.

$$\text{Erase timeout of X AU} = \frac{T_{ERASE}}{N_{ERASE}} * X + T_{OFFSET} \quad (36\text{-}1)$$

**Table 36-37. Erase timeout field**

| ERASE_TIMEOUT | Value Definition |
|---|---|
| 00 | Erase Time-out Calculation is not supported. |
| 01 | 1 [sec] |
| 02 | 2 [sec] |
| 03 | 3 [sec] |
| ..... | ..... |
| 63 | 63 [sec] |

If ERASE_SIZE field is set to 0, this field shall be set to 0.

**ERASE_OFFSET**

This 2-bit field indicates the $T_{OFFSET}$ and one of four values can be selected. This field is meaningless if ERASE_SIZE and ERASE_TIMEOUT fields are set to 0.

**Table 36-38. Erase offset field**

| ERASE_OFFSET | Value Definition |
|---|---|
| 0h | 0 [sec] |
| 1h | 1 [sec] |
| 2h | 2 [sec] |
| 3h | 3 [sec] |

## 36.5. Programming sequence

### 36.5.1. Card identification

The host will be in card identification mode after reset and while it is looking for new cards on the bus. While in card identification mode the host resets all the cards, validates operation voltage range, identifies cards and asks them to publish Relative Card Address (RCA). This operation is done to each card separately on its own CMD line. All data communication in the Card Identification Mode uses the command line (CMD) only.

During the card identification process, the card shall operate in the clock frequency of the identification clock rate $F_{OD}$ (400 kHz).

### Card reset

The command GO_IDLE_STATE (CMD0) is the software reset command and sets *e*•MMC and SD memory card into Idle State regardless of the current card state. The reset command (CMD0) is only used for memory or the memory portion of Combo cards. In order to reset an I/O only card or the I/O portion of a combo card, use CMD52 to write 1 to the RES bit in the CCCR. Cards in Inactive State are not affected by this command.

After power-on by the host, all cards are in Idle State, including the cards that have been in Inactive State before. After power-on or CMD0, all cards' CMD lines are in input mode, waiting for start bit of the next command. The cards are initialized with a default relative card address (RCA) and with a default driver strength with 400 KHz clock frequency.

### Operating voltage range validation

At the start of communication between the host and the card, the host may not know the card supported voltage and the card may not know whether it supports the current supplied voltage. To verify the voltage, the following commands are defined in the related specification.

The SEND_OP_COND (CMD1 for *e*•MMC), SD_SEND_OP_COND (ACMD41 for SD memory), IO_SEND_OP_COND (CMD5 for SD I/O) command is designed to provide hosts with a mechanism to identify and reject cards which do not match the $V_{DD}$ range desired by the host. This is accomplished by the host sending the required $V_{DD}$ voltage window as the operand of this command. If the card cannot perform data transfer in the specified range it must discard itself from further bus operations and go into Inactive State. Otherwise, the card shall respond sending back its $V_{DD}$ range.

If the card can operate on the supplied voltage, the response echoes back the supply voltage and the check pattern that were set in the command argument.

If the card cannot operate on the supplied voltage, it returns no response and stays in idle state. It is mandatory to issue CMD8 prior to ACMD41 to initialize SDHC Card. Receipt of CMD8 makes the cards realize that the host supports the Physical Layer Version 2.00 and the card can enable new functions.

### Card identification process

The card identification process differs in different cards. The card can be of the type *e*•MMC, SD, or SD I/O. All types of SD I/O cards are supported, that is, SDIO_IO_ONLY, SDIO_MEM_ONLY, and SDIO COMBO cards. The identification process sequence includes the following steps:

1. Check if the card is connected.

2. Identify the card type; SD, *e*•MMC, or SD I/O.
– Send CMD5 first. If a response is received, then the card is SD I/O
– If not, send ACMD41; if a response is received, then the card is SD.
– Otherwise, the card is an *e*•MMC.

3. Initialization the card according to the card type.
Use a clock source with a frequency = $F_{OD}$ (that is, 400 KHz) and use the following command sequence:
– SD card - Send CMD0, ACMD41, CMD2, CMD3.
– SDHC card - send CMD0, CMD8, ACMD41, CMD2, CMD3.
– SD I/O - Send CMD52, CMD0, CMD5, if the card doesn't have memory port, send CMD3; otherwise send ACMD41, CMD11 (optional), CMD2, and CMD3.
– *e*•MMC - Send CMD0, CMD1, CMD2, CMD3.

### 36.5.2. Boot operation

If, after power-on or reset (either hardware reset or CMD0 with the argument 0xF0F0F0F0), the CMD line is held low for more than 74 clock cycles before sending the first command (normal boot mode), or CMD0 with the argument 0xFFFFFFFA is sent (alternative boot mode), the card recognizes that the boot mode has been initiated and starts internal boot data preparation.

The partition from which the host will read the boot data can be selected in advance using EXT_CSD byte [179], bits [5:3]. The data size that the host can read during boot operation can be calculated as 128KB × BOOT_SIZE_MULT (EXT_CSD byte [226]).

The host can choose to use single data rate mode with backward-compatible interface timing, SDR with high-speed interface timing or DDR timing (if it supported) by setting a proper value in EXT_CSD register byte [177] bits [4:3]. EXT_CSD register byte [228], bit 2 tells the host if the high-speed timing during boot is supported by the device. EXT_CSD register byte [228], bit 1 tells the host if the dual data rate mode during boot is supported by the device.

HS200 mode is not supported during boot operation.

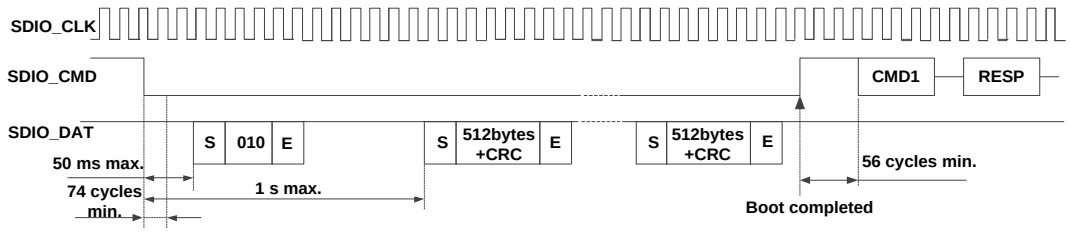
The host can choose to receive boot acknowledge from the card by setting "1" in EXT_CSD register, byte [179], bit 6, so that the host can recognize that the card is operating in boot mode.

If boot acknowledge is enabled, the card has to send acknowledge pattern "010" to the host within 50ms after the CMD line goes low. If boot acknowledge is disabled, the card will not send out acknowledge pattern "010".

#### Normal boot operation

If CMD line is held low lasts for at least 74 clock cycles, the card will recognize that the boot mode is being initiated before issuing the first command. Within one second after the CMD line becomes low, the card starts to run in SDIO_DAT sends the first boot code data online. The host must enable SDIO_CMD line remains low until all boot data is read. The host can connect the SDIO_CMD line is pulled up to terminate the boot mode.

**Figure 36-16. Boot operation timing**




To perform the normal boot process the following steps needed:

1. Reset or power-up the card.

2. If a boot enable ACK, need to enable the ACKEN, Configuration the SDIO_ACKTO and enable the ACKFAIL and ACKTO flag.

3. Enable the data reception by setting DATADIR to 1 and the number of data to be received in DATALEN. Enable the DTTMOUT, DTEND and CMDSEND interrupts for end of command confirmation.

4. Set BOOTMOD to 0, select the normal boot operation mode, and enable boot in BOOTMODEN. Boot acknowledge timeout is enabled and CMD line remains low.

5. The ACKTO or ACKFAIL flag can be used to detect whether the boot acknowledgment is received.
– If the boot acknowledgment is not received in time, the ACKTO flag will appears.
– If an incorrect boot acknowledgment is received, the ACKFAIL flag appears.

6. When all boot data has been received the DTEND flag appears.
– If data CRC fails the DTCRCERR flag will be set.
– If data timeout the DTTMOUT flag will be set.

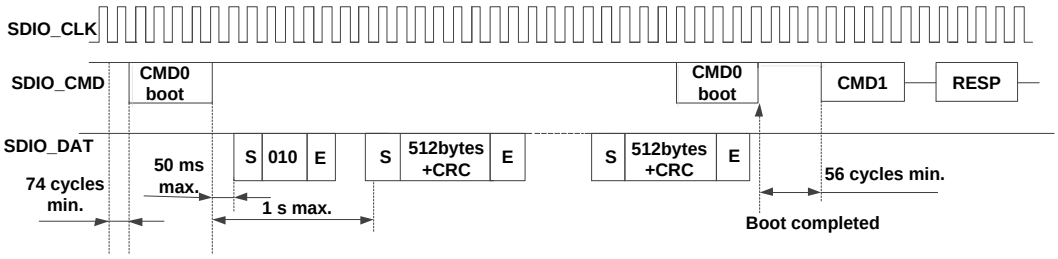
7. When last boot data has been received, read boot data from the FIFO until FIFO is empty (RFE = 1) after which DTEND flag is then set.

8. The boot process can be terminated by clearing BOOTMODEN bit, and the SDIO_CMD line becomes high. After 56 cycles, a CMDSEND flag is generated to indicate that the boot procedure is over and the card is ready to receive a new command.

### Alternative boot operation

After card reset or power-up, if the master send CMD0 (with argument of 0xFFFFFFFA) after 74 clock cycles, before CMD1 is issued, the card recognizes that alternative boot mode is being initiated and starts preparing boot data internally. Within 1 second after the CMD0 (with argument of 0xFFFFFFFA) has been sent, the slave card starts to send the first boot data on the SDIO_DAT line(s). The host can terminate boot mode by issuing CMD0 (with argument of 0xF0F0F0F0).

**Figure 36-17. Alternative Boot operation timing**




The following steps are perform the alternative boot sequence:

1. Reset or power-up the card.

2. If a boot enable ACK, need to enable the ACKEN, set the ACKTIME and enable the ACKFAIL and ACKTO flag.

3. Enable the data reception by setting DATADIR to 1 and the number of data to be received in DATALEN. Enable the DTTMOUT and DTEND flags.

4. Set BOOTMOD to 1, select the alternative boot operation mode, load the CMD0 (with argument of 0xFFFFFFFA) in the command registers. Enable CMDSEND flag for end of command confirmation, enable boot in BOOTMODEN, and set CSMEN to 1. When the command has been sent the CMDSEND flag is generated, and the BOOTMODEN bit must be cleared.

5. The ACKTO or ACKFAIL flag can be used to detect whether the boot acknowledgment is received.
– If the boot acknowledgment is not received in time, the ACKTO flag will appears.
– If an incorrect boot acknowledgment is received, the ACKFAIL flag appears.

6. When all boot data has been received, the DTEND flag appears.
– If data CRC fails the DCRCFAIL flag will be set.
– If data timeout the DTIMEOUT flag will be set.

7. When last boot data has been received, read boot data from the FIFO until FIFO is empty (RFE = 1) after which DTEND flag is then set.

8. The BOOTMODEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with alternative boot.This causes the CMDSEND flag to occur 56 cycles after the command. The CMDSEND flag indicates that the boot procedure is over and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMOD has to be cleared to terminate the boot operation.

### 36.5.3. No data commands

To send any non-data command, the software needs to program the SDIO_CMDCTL register and the SDIO_CMDAGMT register with appropriate parameters. Using these two registers, the host forms the command and sends it to the command bus. The host reflects the errors in the command response through the error bits of the SDIO_STAT register.

When a response is received the host sets the CMDRECV (CRC check passed) or CCRCERR (CRC check error) bit in the SDIO_STAT register. A short response is copied in SDIO_RESP0, while a long response is copied to all four response registers. The SDIO_RESP3 bit 31 represents the MSB, and the SDIO_RESP0 bit 0 represents the LSB of a long response.

### 36.5.4. Single block or multiple block write

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card. The block consists of start bits (1 or 4 bits LOW), data block, CRC and end bits(1 or 4 bits HIGH). If the CRC fails, the card indicates the failure on the SDIO_DAT line and the transferred data are discarded and not written, and all further transmitted blocks are ignored.

If the host uses partial blocks whose accumulated length is not block aligned, block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card will detect the block misalignment error before the beginning of the first misaligned block. The card shall set the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer. The write operation will also be aborted if the host tries to write data on a write protected area. In this case, however, the card will set the WP_VIOLATION bit (in the status register).

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents.

Some cards may require long and unpredictable time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT0 line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY_FOR_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

For SD card. Setting a number of write blocks to be pre-erased (ACMD23) will make a following Multiple Block Write operation faster compared to the same operation without preceding ACMD23. The host will use this command to define how many number of write blocks are going to be send in the next write operation.

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the SDIO_DATALEN register.

2. Write the block size in bytes (BLKSZ) in the SDIO_DATACTL register; the host sends data in blocks of size BLKSZ.

3. Program SDIO_CMDAGMT register with the data address to which data should be written.

4. Program the SDIO_CMDCTL register. For SD memory and *e*•MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SD I/O cards, use CMD53 for both single-block and multiple-block transfers.

5. Write data to SDIO_FIFO.

6. Software should look for data error interrupts. If required, software can terminate the data transfer by sending the STOP command (CMD12).

7. When a DTEND interrupt is received, the data transfer is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the host should send the STOP command.

### 36.5.5. Single block or multiple block read

Block read is block oriented data transfer. The basic unit of data transfer is a block whose maximum size is defined in the CSD (READ_BL_LEN), it is always 512 bytes. If READ_BL_PARTIAL(in the CSD) is set, smaller blocks whose starting and ending address are entirely contained within 512 bytes boundary may be transmitted.

CMD17 (READ_SINGLE_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ_MULTIPLE_BLOCK) starts a transfer of several consecutive blocks. CRC is appended to the end of each block, ensuring data transfer integrity.

Block Length set by CMD16 can be set up to 512 bytes regardless of READ_BL_LEN.

Blocks will be continuously transferred until a STOP_TRANSMISSION command (CMD12) is issued. The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

When the last block of user area is read using CMD18, the host should ignore OUT_OF_RANGE error that may occur even the sequence is correct.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first misaligned block, set the ADDRESS_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.

Steps involved in a single block or multiple block read are:

1. Write the data size in bytes in the SDIO_DATALEN register.

2. Write the block size in bytes (BLKSZ) in the SDIO_DATACTL register. The host expects data from the card in blocks of size BLKSZ each.

3. Program the SDIO_CMDAGMT register with the data address of the beginning of a data read.

4. Program the SDIO_CMDCTL. For SD and *e*•MMC cards, using CMD17 for a single-block read and CMD18 for a multiple-block read. For SD I/O cards, using CMD53 for both single-block and multiple-block transfers. After writing to the CMD register, the host starts executing the command, when the command is sent to the bus, the CMDRECV flag is set.

5. Software should look for data error interrupts. If required, software can terminate the data transfer by sending a STOP command.

6. The software should read data from the FIFO and make space in the FIFO for receiving more data.

7. When a DTEND interrupt is received, the software should read the remaining data in the FIFO.

### 36.5.6. Stream write and stream read (e•MMC only)

#### Stream write

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. If partial blocks are allowed (if CSD parameter WRITE_BL_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. Since the amount of data to be transferred is not determined in advance, CRC cannot be used.

If the host provides an out of range address as an argument to CMD20, the card will reject the command, remain in Tran state and respond with the ADDRESS_OUT_OF_RANGE bit set.

Note that the stream write command works only on a 1 bit bus configuration (on DAT0). If CMD20 is issued in other bus configurations, it is regarded as an illegal command.

In order to sustain data transfer in stream mode of the card, the time it takes to receive the data (defined by the bus clock rate) must be less than the time it takes to program it into the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for the stream write operation is given by the following formula:

$$\text{max write frequence} = \min\left(\text{TRAN_SPEED}, \frac{8*2^{\text{WRITE_BL_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W_FACTOR}}\right) \quad (36\text{-}2)$$

**TRAN_SPEED**: Max bus clock frequency.

**WRITE_BL_LEN**: Max write data block length.

**NSAC**: Data read access-time 2 in CLK cycles.

**TAAC**: Data read access-time 1.

**R2W_FACTOR**: Write speed factor.

All the parameters are defined in CSD register. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, and while ignoring

all further data transfer, wait (in the Receive-data-State) for a stop command. As the host sends CMD12, the card will respond with the TXURE bit set and return to Transfer state

#### Stream read

There is a stream oriented data transfer controlled by READ_DAT_UNTIL_STOP (CMD11). This command instructs the card to send its data, starting at a specified address, until the host sends a STOP_TRANSMISSION command (CMD12). The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

If the host provides an out of range address as an argument to CMD11, the card will reject the command, remain in Transfer state and respond with the ADDRESS_OUT_OF_RANGE bit set.

Note that the stream read command works only on a 1 bit bus configuration (on DAT0). If CMD11 is issued in other bus configurations, it is regarded as an illegal command.

If the end of the memory range is reached while sending data, and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined. As the host sends CMD12 the card will respond with the ADDRESS_OUT_OF_RANGE bit set and return to Tran state.

In order to sustain data transfer in stream mode of the card, the time it takes to transmit the data (defined by the bus clock rate) must be less than the time it takes to read it out of the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for stream read operation is given by the following formula:

$$\text{max read frequence} = \min\left(\text{TRAN_SPEED}, \frac{8*2^{\text{READ_BL_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W_FACTOR}}\right) \quad (36\text{-}3)$$

**TRAN_SPEED**: Max bus clock frequency.

**READ_BL_LEN**: Max read data block length.

**NSAC**: Data read access-time 2 in CLK cycles.

**TAAC**: Data read access-time 1.

**R2W_FACTOR**: Write speed factor.

All the parameters are defined in CSD register. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, and while ignoring all further data transfer, wait (in the Receive-data-State) for a stop command. As the host sends CMD12, the card will respond with the RXORE bit set and return to Transfer state

### 36.5.7. Erase

The erasable unit of the *e*•MMC/SD memory is the “Erase Group”; Erase group is measured in write blocks which are the basic writable units of the card. The size of the Erase Group is a card specific parameter and defined in the CSD.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three steps sequence. First the host defines the start address of the range using the ERASE_GROUP_START (CMD35)/ERASE_WR_BLK_START(CMD32) command, next it defines the last address of the range using the ERASE_GROUP_END (CMD36)/ERASE_WR_BLK_END(CMD33) command and finally it starts the erase process by issuing the ERASE (CMD38) command. The address field in the erase commands is an Erase Group address in byte units. The card will ignore all LSB's below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command (CMD35, CMD36, and CMD38) is received out of the defined erase sequence, the card shall set the ERASE_SEQ_ERROR bit in the status register and reset the whole sequence.

If the host provides an out of range address as an argument to CMD35 or CMD36, the card will reject the command, respond with the ADDRESS_OUT_OF_RANGE bit set and reset the whole erase sequence.

If an 'non erase' command (neither of CMD35, CMD36, CMD38 or CMD13) is received, the card shall respond with the ERASE_RESET bit set, reset the erase sequence and execute the last command.

If the erase range includes write protected blocks, they shall be left intact and only the non-protected blocks shall be erased. The WP_ERASE_SKIP status bit in the status register shall be set.

As described above for block write, the card will indicate that an erase is in progress by holding DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

### 36.5.8. Bus width selection

After the host has verified the functional pins on the bus it should change the bus width configuration.

For *e*•MMC, using the SWITCH command (CMD6).The bus width configuration is changed by writing to the BUS_WIDTH byte in the Modes Segment of the EXT_CSD register (using the SWITCH command to do so). After power-on or software reset, the contents of the BUS_WIDTH byte is 0x00. If the host tries to write an invalid value, the BUS_WIDTH byte is not changed and the SWITCH_ERROR bit is set. This register is write only.

For SD memory, using SET_BUS_WIDTH command (ACMD6) to change the bus width. The default bus width after power up or GO_IDLE_STATE command (CMD0) is 1 bit. SET_BUS_WIDTH (ACMD6) is only valid in a transfer state, which means that the bus width can be changed only after a card is selected by SELECT/DESELECT_CARD (CMD7).

### 36.5.9. Protection management

In order to allow the host to protect data against erase or write, three methods for the cards

are supported in the card:

**CSD register for card protection (optional)**

The entire card may be write protected by setting the permanent or temporary write protect bits in the CSD. Some cards support write protection of groups of sectors by setting the WP_GRP_ENABLE bit in the CSD. It is defined in units of WP_GRP_SIZE erase groups as specified in the CSD. The SET_WRITE_PROT command sets the write protection of the addressed write protected group, and the CLR_WRITE_PROT command clears the write protection of the addressed write protected group.

The High Capacity SD Memory Card does not support Write Protection and does not respond to write protection commands (CMD28, CMD29 and CMD30).

**Write protect switch on the card (SD memory and SD I/O card)**

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open it means that the card is write protected. If the window is closed the card is not write protected.

**Password card Lock/Unlock Operation**

The Password Card Lock/Unlock protection is described in ***Card Lock/Unlock operation***.

### 36.5.10. Card Lock/Unlock operation

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size are kept in a 128-bit PWD and 8-bit PWD_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0), ACMD41, CMD16 and lock card command class (class 7). Thus, the host is allowed to reset, initialize, select, query for status, but not to access data on the card. If the password was previously set (the value of PWD_LEN is not 0), the card will be locked automatically after power on.

Similar to the existing CSD register write commands, the lock/unlock command is available in "transfer state" only. This means that it does not include an address argument and the card shall be selected before using it.

The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). ***Table 36-39. Lock card data structure*** describes the structure of the command data block.

**Table 36-39. Lock card data structure**

| Byte | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved(all set to 0) | | | | ERASE | LOCK_UNLOCK | CLR_PWD | SET_PWD |

| | |
|---|---|
| 1 | PWDS_LEN |
| 2 | Password data(PWD) |
| …… | |
| PWDS_LEN+1 | |

**ERASE**: 1 Defines Forced Erase Operation. In byte 0, bit 3 will be set to 1 (all other bits shall be 0). All other bytes of this command will be ignored by the card.

**LOCK/UNLOCK**: 1 = Locks the card. 0 = Unlock the card (note that it is valid to set this bit together with SET_PWD but it is not allowed to set it together with CLR_PWD).

**CLR_PWD**: 1 = Clears PWD.

**SET_PWD**: 1 = Set new password to PWD.

**PWDS_LEN**: Defines the following password(s) length (in bytes). In case of a password change, this field includes the total password length of old and new passwords. The password length is up to 16 bytes. In case of a password change, the total length of the old password and the new password can be up to 32 bytes.

**Password data**: In case of setting a new password, it contains the new password. In case of a password change, it contains the old password followed by the new password.

### Setting the password

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the new password. In the case that a password replacement is done, then the block size shall consider that both passwords (the old and the new one) are sent with the command.
- Send the Card Lock/Unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode (SET_PWD), the length (PWDS_LEN) and the password itself. In the case that a password replacement is done, then the length value (PWDS_LEN) shall include both passwords (the old and the new one) and the password data field shall include the old password (currently used) followed by the new password. Note that the card shall handle the calculation of the new password length internally by subtracting the old password length from PWDS_LEN field.
- In the case that the sent old password is not correct (not equal in size and content), then the LOCK_UNLOCK_FAILED error bit will be set in the status register and the old password does not change. In the case that the sent old password is correct (equal in size and content), then the given new password and its size will be saved in the PWD and PWD_LEN registers, respectively.

### Reset the password

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line

including the 16-bit CRC. The data block shall indicate the mode CLR_PWD, the length (PWDS_LEN), and the password itself. If the PWD and PWD_LEN content match the sent password and its size, then the content of the PWD register is cleared and PWD_LEN is set to 0. If the password is not correct, then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

### Locking a card

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWDS_LEN) and the password itself.

If the PWD content is equal to the sent password, then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct, then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

### Unlocking the card

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWDS_LEN) and the password itself.

If the PWD content is equal to the sent password, then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct, then the LOCK_UNLOCK_FAILED error bit will be set in the status register.

## 36.5.11. Sleep

The *e*•MMC card can switch between Sleep state and Standby state through CMD5. In the sleep state, the power consumption of the card is minimized, and the Vcc power can be turned off at this time.

– Sleep command: The bit 15 as set to 1 in CMD5 argument.
– Awake command: The bit 15 as set to 0 in CMD5 argument.

Use Sleep command to start the state transition from Standby state to Sleep state. The card indicates the transition phase busy by pulling down the SDIO_DAT0 line. No further commands should be sent during the busy. When the device stops pulling down the SDIO_DAT0 line, indicates reach Sleep state.

Use Awake command to start the state transition from Sleep state to Standby state. The card indicates indicates the transition phase busy by pulling down the SDIO_DAT0 line. No further commands should be sent during the busy. When the device stops pulling down the

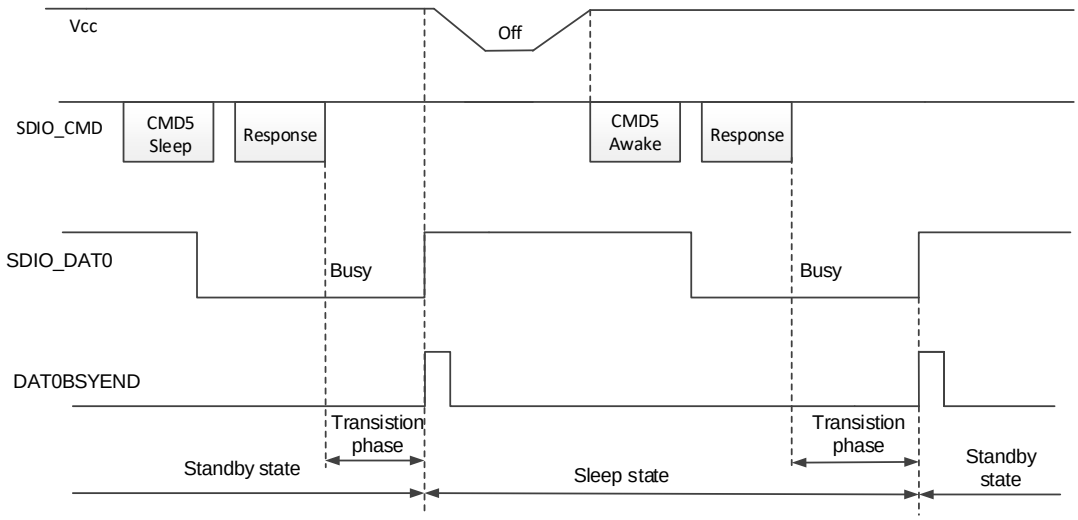
SDIO_DAT0 line, indicates reach Standby state.

To set the card to sleep, follow these steps:
1. Enable DAT0BSYEND interrupt.
2. Send CMD5 (Sleep command).
3. When DAT0BSYEND is interrupted, the card is in Sleep state.
4. It is allowed to turn off the Vcc power.

To set the card to Standby, follow these steps:
1. Turn on the Vcc power supply and wait for it to reach the minimum working voltage.
2. Enable DAT0BSYEND interrupt.
3. Send CMD5 (Awake command).
4. When DAT0BSYEND interrupt occurs, the card is in Standby mode.

During the Sleep state the Vcc power supply may be switched off. This is to enable even further system power consumption saving. The Vcc supply is allowed to be switched off only after the Sleep state has been reached (the card has stopped to pull down the SDIO_DAT0 line). The Vcc supply have to be ramped back up at least to the minimum working voltage level before the state transition from Sleep state to Standby state is allowed to be initiated (Awake command).

**Figure 36-18. CMD5 timing**




## 36.5.12. CMD12 send timing

CMD12 (stop transmission command) is used to stop or abort the data transmission. The card data transmission is terminated after 2 clock cycles after the end bit of a CMD12.

All data read and write commands can be terminated at any time by a CMD12. If data transmission is in progress, the CMD12 send sequence includes the following steps:

1. Configure the CMD12 command in registers and set the TRSTOP register. When DSM receives the command, it will generate CSM stop signal.

2. Clear WAITDEND register.

3. When IDMAEN = 0, set FIFOREST bit to 1.
– Host sends data, when the CMDRECV flag appears, the firmware will stop transfer data to the FIFO. Subsequently set FIFOREST bit to 1 and flushes the FIFO.
– Host receives data, when the CMDRECV flag appears, the firmware will read the remaining data from the the FIFO. Subsequently set FIFOREST bit to 1.

4. When IDMAEN = 1, hardware operates the FIFO.
– Host sends data, when the Abort flag appears, the hardware will stop IDMA. Subsequently flushes the FIFO.
– Host receives data, when the Abort flag appears, the hardware uses IDMA to transfer the remaining FIFO data to RAM.

5. When FIFO is empty/reset, the DATABOR flag is generated

**Table 36-40. CMD12 instructions**

| Data operation type | CMD12 Description |
|---|---|
| Multiple block write with pre-defined block count | The device will receives the requested number of data blocks, terminate the transaction and return to TRANSFER state. Stop command is not required at the end of this type of multiple block write, unless terminated with an error. |
| Multiple block read with pre-defined block count | The card will transfer the requested number of data blocks, terminate the transaction and return to TRANSFER state. Stop command is not required at the end of this type of multiple block read, unless terminated with an error. |
| Open-ended Multiple block write | The number of blocks for the write multiple block operation is not defined. The device will continuously receive and program data blocks until a stop transmission command is received. |
| Open-ended Multiple block read | The number of blocks for the read multiple block operation is not defined. The Device will continuously transfer data blocks until a stop transmission command is received |
| stream write | The data flow is terminated by CMD12. |
| stream read | The data flow is terminated by CMD12. |

### CMD12 affect block operation

To stop block transmission at the end of data, the end bit of CMD12 should be sent after the end bit of last block.

When data is written to the card, the CMD12 end bit is sent after the CRC token end bit of the data block is written. This requires the CMD12 transmission process to follow the block transmission timing.

To stop the Open-ended Multiple block write operation, follow these steps：

1. Before starting data transmission, set TRANSMOD[1:0] to "11".

2. Wait for DTEND to be set, the data sent by DSM will not exceed SDIO_DATALEN.

3. CSM sents CMD12. The card is set to idle state.

When reading data from the card, the end bit of CMD12 should be sent as early as possible. That is, when the card reads the last bit of the data block.

To stop the Open-ended Multiple block read operation, follow these steps：

1. Before starting data transmission, set TRANSMOD[1:0] to "11".

2. Wait for DTEND to be set, even if the card is sending more data, the data received by DSM will not be greater than SDIO_DATALEN.

3. CSM sents CMD12. The card sending data is stopped and the card is set to idle state.

### CMD12 affect stream operation

To stop stream transmission after the last byte to be transmitted, the CMD12 end bit timing should be sent at the end of the last byte of the data stream. Follow these steps to write stream data:

1. Initialize DSM, set TRANSMOD[1:0] to "10".

2. Set TREN to 1 and sent stream write command.

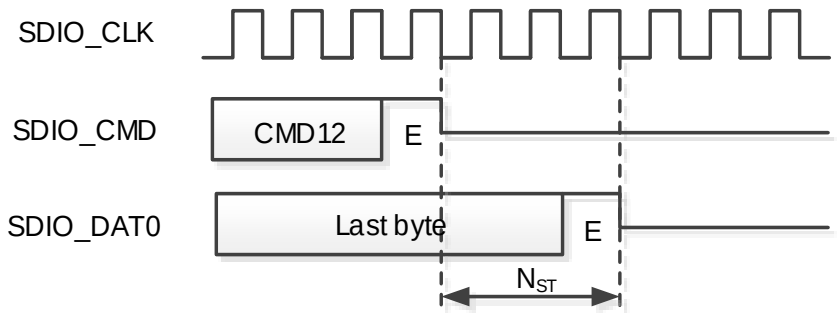
3. Preload CMD12 in command registers with the TRSTOP bit set.

4. Configure the CSM to send a command only after WAITDEND = 1 end of last data of DATALEN.

5. CSM sents CMD12. the stream data end bit and command end bit are aligned.
– If DATALEN > 5, CMD12 is waited in the CSM to be aligned with the data transfer end bit.
– If DATALEN < 5, CMD12 will start early and DSM will keep WaitS state, to align the data transfer end bit with the CMD12 end bit.

6. The stream data writing process can be stopped at any time by clearing the WAITDEND bit. This will cause the preloaded CMD12 to be sent immediately and stop the stream writing process.

**Figure 36-19. CMD12 stream timing**




To stop the stream transfer read process after the last byte, the CMD12 end bit timing should be sent after the last byte of the data stream.

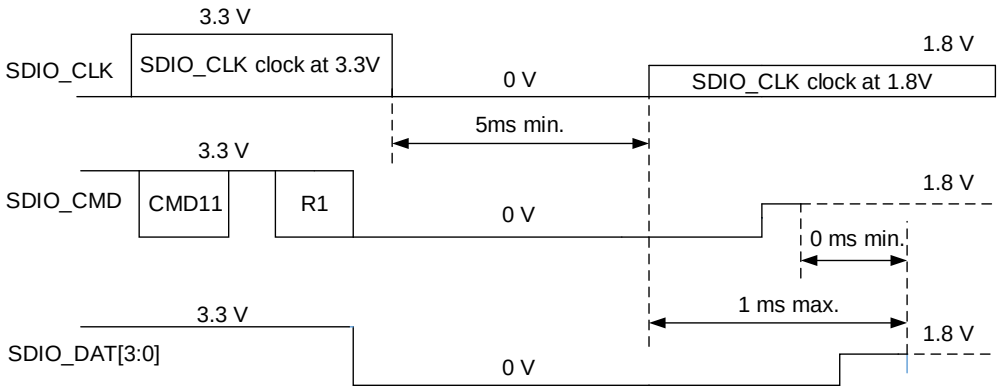
**Note:** 1. Wait for DTEND = 1 (data receiving completed) before sending the CMD12. Even if the card is sending more data. the data received by DSM will not be greater than SDIO_DATALEN. 2. The SDIO does not receive any more data when DATACNT = 0, even when the card continues sending data.

## 36.6. Specific operations

### 36.6.1. UHS-I

Ultra High Speed Bus Speed Mode (SDR12, SDR25, SDR50, DDR50 and SDR104) phase I Voltage Switch (UHS-1) mode requires the support voltage switching from 3.3V to 1.8V. The default voltage is 3.3V when the card is powered on. A signal voltage switch sequence is started by this CMD11. When the voltage sequence is completed successfully, the card will enter the UHS-I mode with the default SDR12, and the input and output timings of the card will change.

**Figure 36-20. CMD11 voltage switch sequence**




The following steps are perform the signal Voltage Switch sequence:

1. The SDIO_CLK clock frequency must be set in the range 100 kHz - 400 kHz, before starting of Voltage Switch operation.

2. The host sets VSEN = 1 for starts the Voltage Switch, and then sends the CMD11.

3. Host receiving card R1 response.
– If the response CRC passes, the Voltage Switch process will continue and the host will no longer drive the CMD and SDIO_DAT[3:0] signals until the voltage switching procedure is completed. After several cycles of response, SDIO_CLK will stop and CLKSTOP flag will be set to 1.
– If the response CRC fails (CCRCERR = 1) or the response timeout (CMDTMOUT = 1), the Voltage Switch procedure is stopped.

4. At the next clock after the R1 response, the card drives CMD and SDIO_DAT[3:0] to low.

5. After receiving the R1 response, the host monitors the SDIO_DAT0 line by reading the DAT0BSY register. The SDIO_DAT0 line is sampled after 2 SDIO_CLK clock cycles following

the response. The Firmware may read the DAT0BSY register bit following the CLKSTOP flag.
– When DAT0BSY is detected as low level, the firmware switches the voltage regulator to 1.8V. After that, it instructs the SDIO to start timing the critical section of the Voltage Switch sequence by setting register bit VSSTART to 1. The hardware continues to stop the SDIO_CLK by holding it low for at least 5 ms.
– When DAT0BSY is detected as high level, the firmware will abort the Voltage Switch sequence and cycle power the device card.

6. When SDIO_CLK is low, the card will start to switch the signal voltage to 1.8V.

7. The SDIO hardware will restart the SDIO_CLK at least 5 ms later.

8. Within 1 ms after SDIO_CLK switch is detected, the card will drive the CMD and DAT[3:0] to high level for at least 1 clock cycle, and then stop driving the CMD and DAT[3:0].

9. After 1 ms after the SDIO_CLK is restarted, the SDIO hardware will sample SDIO_DAT0 into DAT0BSY and generate VSEND flag.

10. If VSEND flag is set, the host needs to read DAT0BSY register to judge SDIO_DAT0 line, to confirm whether the voltage switching procedure is completed:
– If DAT0BSY is high, Voltage Switch has been completed successfully.
– If DAT0BSY is low, Voltage Switch has failed, the host cycles the card power.

## 36.7. Register definition

SDIO0 base address: 0x5200 7000

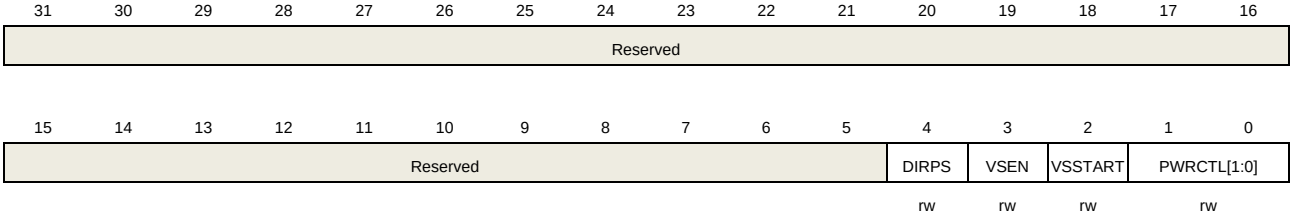
SDIO1 base address: 0x4802 2400

### 36.7.1. Power control register (SDIO_PWRCTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4 | DIRPS | Command and data direction polarity selection bit<br>This bit can only be configured when the PWRCTL[1:0] bits are cleared.<br>0: Direction signal is low, the voltage transceiver IOs driven as output.<br>1: Direction signal is high, the voltage transceiver IOs driven as output. |
| 3 | VSEN | Voltage switch enable bit<br>This bit is used to stop the SDIO_CLK after the voltage switch command response.<br>This bit can only be configured by firmware when CSMEN bit is cleared.<br>0: SDIO_CLK clock kept unchanged, after successfully received command response.<br>1: SDIO_CLK clock stopped, after successfully received command response. |
| 2 | VSSTART | Voltage switch start bit<br>This bit is used to start the timing of the voltage switch sequence.<br>0: Voltage switch sequence not active and not started<br>1: Voltage switch sequence active or started |
| 1:0 | PWRCTL[1:0] | SDIO power control bits<br>These bits control the SDIO state, card input or output.<br>These bits can only be configured when the SDIO is in the power down state.<br>00: State after reset (reset: SDIO is disabled, clock is stoped, SDIO_CMD and SDIO_DAT are HiZ state, and SDIO_CLK is driven low), if written 00, SDIO power off (power off: SDIO is disabled, clock is stoped, SDIO_CLK, SDIO_CMD, SDIO_DAT are driven high).<br>01: Reserved (When written 01, PWRCTRL value does not change) |

10: Power-cycle, the SDIO is disabled and the clock to the card is stopped, SDIO_DAT, SDIO_CMD and SDIO_CLK are driven low.

11: SDIO Power on. After power on, SDIO enables have 74 SDIO_CLK cycles delay, any further write is ignored, PWRCTL[1:0] value keeps 11.

## 36.7.2. Clock control register (SDIO_CLKCTL)

Address offset: 0x04

Reset value: 0x0000 0000

This register controls the output clock SDIO_CLK.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | RCLK[1:0] | | BUSSP | DRSEL | HWEN | CLKEDGE |
| rw | | | | | | | | | | rw | | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BUSMODE[1:0] | | Reserved | CLKPWR SAV | Reserved | | DIV[9:0] | | | | | | | | | |
| rw | | | rw | | | rw | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:20 | RCLK[1:0] | Receive clock selection bits<br>These bits can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>00: select SDIO_IN_CLK clock<br>01: select SDIO_CLKIN clock<br>10: select SDIO_FB_CLK clock<br>11: Reserved (select SDIO_IN_CLK clock) |
| 19 | BUSSP | Bus speed mode selection bit<br>This bit can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>0: DS, HS, SDR12, SDR25 bus speed.<br>1: SDR50, DDR50, SDR104 bus speed. |
| 18 | DRSEL | Data rate selection bit<br>This bit can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>0: SDR selected<br>1: DDR selected |
| 17 | HWEN | Hardware flow control enable bit<br>If this bit is set, the meaning of the TFF and RFF flags will change.<br>This bit can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>0: HW flow control is disabled<br>1: HW flow control is enabled |
| 16 | CLKEDGE | Command and data SDIO_CLK dephasing selection bit |

| | | |
|---|---|---|
| | | These bits can only be written when the (DATSTA = 0) and (CMDSTA = 0).<br>If DIV = 0, this bit has no effect. Data and Command change on SDIO_CLK falling edge.<br>0: if DIV > 0 and DRSEL = 0, command and data changed on the same CK_SDIO falling edge succeeding the rising edge of SDIO_CLK. SDIO_CLK edge occurs on CK_SDIO rising edge.<br>if DIV > 0 and DRSEL = 1, command changed on the CK_SDIO falling edge succeeding the rising edge of SDIO_CLK. Data changed on the CK_SDIO falling edge succeeding a SDIO_CLK edge. SDIO_CLK edge occurs on CK_SDIO rising edge.<br>1: if DIV > 0 and DRSEL = 0, command and data changed on the same CK_SDIO rising edge generating the SDIO_CLK falling edge. SDIO_CLK edge occurs on CK_SDIO rising edge.<br>if DIV > 0 and DRSEL= 1, command changed on the same CK_SDIO rising edge generating the SDIO_CLK falling edge. Data changed on the SDIO_CLK falling edge succeeding a SDIO_CLK edge. SDIO_CLK edge occurs on CK_SDIO rising edge. |
| 15:14 | BUSMODE[1:0] | SDIO card bus mode control bit<br>These bits can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>00: 1-bit SDIO card bus mode selected (default and SDR only), SDIO_DAT0.<br>01: 4-bit SDIO card bus mode selected, SDIO_DAT [3:0].<br>10: 8-bit SDIO card bus mode selected, SDIO_DAT[7:0].<br>11: No effect |
| 13 | Reserved | Must be kept at reset value. |
| 12 | CLKPWRSAV | SDIO_CLK clock dynamic switch on/off for power saving.<br>This bit can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>This bit controls SDIO_CLK clock dynamic switch on/off when the bus is idle for power saving.<br>0: SDIO_CLK clock is always on<br>1: SDIO_CLK closed when bus idle |
| 11:10 | Reserved | Must be kept at reset value. |
| 9:0 | DIV[9:0] | Clock division<br>These bits can only be configured when the (DATSTA = 0) and (CMDSTA = 0).<br>This field defines the division factor to generator SDIO_CLK clock to card. The SDIO_CLK is divider from CK_SDIO, and the SDIO_CLK frequency = CK_SDIO / (DIV[9:0] * 2).<br>0x000: SDIO_CLK = CK_SDIO / 1 (Does not support DDR)<br>0x001: SDIO_CLK = CK_SDIO / 2<br>0x002: SDIO_CLK = CK_SDIO / 4<br>……<br>0x3FF: SDIO_CLK = CK_SDIO / 2046 |

**Note:** When card is in identification mode, the frequency of SDIO_CLK must be less than 400 kHz.

If the RCA are assigned to all cards, you can change the clock frequency to the maximum card bus frequency.

Between Two write accesses to this register, it needs at least 7 HCLK periods which used to sync the registers to SDIO_CLK clock domain.

### 36.7.3. Command argument register (SDIO_CMDAGMT)

Address offset: 0x08
Reset value: 0x0000 0000

This register defines 32bit command argument, which will be used as part of the command (bit 39 to bit 8).

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMDAGMT[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMDAGMT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CMDAGMT[31:0] | SDIO card command argument<br>This field defines the card command argument which sent to card. This field is the bits [39:8] of command message. If the command message contains an argument, this field must update before writing SDIO_CMDCTL register when sending a command.<br>These bits can only be configured by firmware when CSMEN = 0. |

### 36.7.4. Command control register (SDIO_CMDCTL)

Address offset: 0x0C
Reset value: 0x0000 0000

The SDIO_CMDCTL register contains the command index and other command control bits to control command state machine.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | CMDSR |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOOTMODEN | BOOTMOD | HOLD | CSMEN | WAITDEND | INTWAIT | CMDRESP[1:0] | | TRSTOP | TREN | CMDIDX[5:0] | | | | | |
| rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | CMDSR | Suspend or Resume command and signals interrupt period start/end bit.<br>This bit can only be configured by firmware when the CSMEN = 0.<br>0: No effect.<br>1: If TREN = 0, the CSM treats the command as a Suspend command.<br>If TREN = 1, the CSM treats the command as a Resume command. |
| 15 | BOOTMODEN | Boot mode procedure enable bit<br>0: Boot disable<br>1: Boot enable |
| 14 | BOOTMOD | Boot mode selection bit<br>This bit can only be configured by firmware when the CSMEN = 0.<br>0: Normal boot mode<br>1: Alternative boot mode |
| 13 | HOLD | Hold DSM transmission and reception of new data block<br>If HOLD bit is set, the DSM does not move from the DS_WaitS state to the DS_Send state or from the DS_WaitR state to the DS_Receive state.<br>This bit can only be configured by firmware when the CSMEN = 0.<br>0: No effect<br>1: Hold transmission and reception data |
| 12 | CSMEN | Command state machine (CSM) enable bit<br>This bit can only be configured by firmware. It is cleared by hardware.<br>0: Command state machine disable (stay on CS_Idle)<br>1: Command state machine enable |
| 11 | WAITDEND | Waits for ends of data transfer.<br>If this bit is set, the command state machine starts to send a command must wait the end of data transfer.<br>0: No effect<br>1: Wait the end of data transfer |
| 10 | INTWAIT | Interrupt wait request<br>If this bit is cleared in Wait state, the interrupt mode is aborted.<br>0: No effect.<br>1: No command timeout and wait interrupt while the command response. |
| 9:8 | CMDRESP[1:0] | Waits command response type bits<br>These bits define the response type after sending a command message.<br>These bits can only be written by firmware when the CSMEN = 0.<br>00: No response<br>01: Short response<br>10: Short response (No CRC) |

| | | |
|---|---|---|
| | | 11: Long response |
| 7 | TRSTOP | Data transfer stop command mode enable bit<br>Transfer stop command mode (CSM treats the command as a data stop transfer command).<br>This bit can only be configured by firmware when the CSMEN = 0.<br>0: No effect<br>1: If command is sent, enable transfer stop command mode and stops DSM data transfer. |
| 6 | TREN | Data transfer command mode enable bit<br>Transfer command mode (CSM treats the command as a data transfer command).<br>This bit can only be configured by firmware when the CSMEN = 0.<br>0: No effect<br>1: If command is sent, enable transfer command mode and stops the interrupt period. |
| 5:0 | CMDIDX[5:0] | Command index<br>These bits can only be configured by firmware when the CSMEN = 0.<br>This field defines the command index to be sent to card. |

**Note:** Between Two write accesses to this register, it needs at least seven HCLK clock periods which used to sync the registers to SDIO_CLK clock domain.

The e•MMC can send two kinds of response: short responses, 48 bits, or long responses,136 bits.

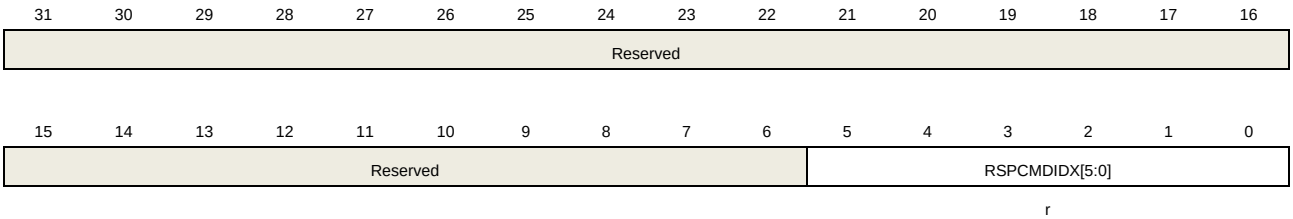
SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

### 36.7.5. Command index response register (SDIO_RSPCMDIDX)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | RSPCMDIDX[5:0] | Last response command index |

Read-only bits field. This field contains the command index of the last command response received. If the response doesn't have the command index (long response and short response of R3), the content of this register is undefined.

## 36.7.6. Response register (SDIO_RESPx x = 0..3)

Address offset: 0x14+(4*x), x = 0..3
Reset value: 0x0000 0000

These register contains the content of the last card response received.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESPx[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESPx[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RESPx[31:0] | Card state. The content of the response, see ***Table 36-41. SDIO_RESPx register at different response type.*** |

The short response is 32 bits, the long response is 127 bits (bit 128 is the end bit 0).

**Table 36-41. SDIO_RESPx register at different response type**

| Register | Short response | Long response |
|---|---|---|
| SDIO_RESP0 | Card response[31:0] | Card response[127:96] |
| SDIO_RESP1 | reserved | Card response [95:64] |
| SDIO_RESP2 | reserved | Card response [63:32] |
| SDIO_RESP3 | reserved | Card response [31:1],plus bit 0 |

## 36.7.7. Data timeout register (SDIO_DATATO)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATATO[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATATO[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:0 | DATATO[31:0] | Data timeout period<br>These bits define the data timeout period count by SDIO_CLK. When the DSM enter the state WaitR or Busy, the internal counter which loads from this register starts decrement. The DSM timeout and enter the state Idle and set the DTTMOUT flag when the counter decreases to 0.<br>These bits can only be written when the (DATSTA = 0) and (CMDSTA = 0). |

**Note:** The data timeout register and the data length register must be updated before being written to the data control register when need a data transfer.

## 36.7.8. Data length register (SDIO_DATALEN)

Address offset: 0x28
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | DATALEN[24:16] | | | | | | | | |
| | | | | | | | rw | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATALEN[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24:0 | DATALEN[24:0] | Data transfer length<br>This register defined the number of bytes to be transferred. When the data transfer starts, the data counter loads this register and starts decrement.<br>These bits can only be written when the DATSTA = 0.<br>If DRSEL = 1, DATALEN is truncated to a multiple of 2. Indicates that the last data odd byte is not transferred.<br>If DATALEN = 0, no data byte are transfered, when requested by a CSMEN and TREN = 1 also no command is transfered. DATAEN and CSMEN are cleared. |

**Note:** If block data transfer selected, the content of this register must be a multiple of the block size (refer to SDIO_DATACTL). The data timeout register and the data length register must be updated before being written to the data control register when need a data transfer.

For multibyte transfer the value in the data length register must be between 1 and 512.

## 36.7.9. Data control register (SDIO_DATACTL)

Address offset: 0x2C
Reset value: 0x0000 0000

This register controls the DSM.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | FIFOREST | ACKEN | Reserved | | | | BLKSZ[3:0] | | | | TRANSMOD[1:0] | | DATADIR | DATAEN |
| | | rw | rw | | | | | rw | | | | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | FIFOREST | FIFO buffer reset, flush all data<br>This bit can only be configured by firmware when IDMAEN = 0 and DATSTA = 1. If a transfer error or transfer hold occurs, this bit takes effect. And automatically cleared by hardware when DATSTA = 0.<br>0: No effect<br>1: Flush any remaining data and reset the FIFO pointers |
| 12 | ACKEN | Boot acknowledgment enable bit<br>This bit can only be configured by firmware when DATSTA = 0.<br>0: Boot acknowledgment disabled<br>1: Boot acknowledgment enabled |
| 11:8 | Reserved | Must be kept at reset value. |
| 7:4 | BLKSZ[3:0] | Data block size<br>These bits defined the block size when data transfer is block transfer.<br>These bits can only be written by firmware when DATSTA = 0.<br>0000: block size = $2^0$ = 1 byte<br>0001: block size = $2^1$ = 2 bytes<br>0010: block size = $2^2$ = 4 bytes<br>0011: block size = $2^3$ = 8 bytes<br>0100: block size = $2^4$ = 16 bytes<br>0101: block size = $2^5$ = 32 bytes<br>0110: block size = $2^6$ = 64 bytes<br>0111: block size = $2^7$ = 128 bytes<br>1000: block size = $2^8$ = 256 bytes<br>1001: block size = $2^9$ = 512 bytes<br>1010: block size = $2^{10}$ = 1024 bytes<br>1011: block size = $2^{11}$ = 2048 bytes<br>1100: block size = $2^{12}$ = 4096 bytes<br>1101: block size = $2^{13}$ = 8192 bytes<br>1110: block size = $2^{14}$ = 16384 bytes<br>1111: reserved<br>When DATALEN is not a multiple of BLKSZ, the transfered data is truncated at a multiple of BLKSZ. |

| | | |
|---|---|---|
| | | If DRSEL = 1, BLKSZ = 0000 must no data are transferred. |
| 3:2 | TRANSMOD[1:0] | Data transfer mode<br>These bits can only be written by firmware when DATSTA = 0.<br>00: Block count data transfer<br>01: Multibyte data transfer (only SD/SD I/O)<br>10: Stream transfer (only *e*•MMC)<br>11: Transfer ends with CMD12 |
| 1 | DATADIR | Data transfer direction<br>This bit can only be configured by firmware when DATSTA = 0.<br>0: Write data to card.<br>1: Read data from card. |
| 0 | DATAEN | Data transfer enable bit<br>This bit is only used to transfer data when the associated data transfer command is not used.<br>This bit can only be configured by firmware when DATSTA = 0.<br>This bit is cleared by hardware when data transfer completes.<br>0: No effect<br>1: Start data transfer without CSM |

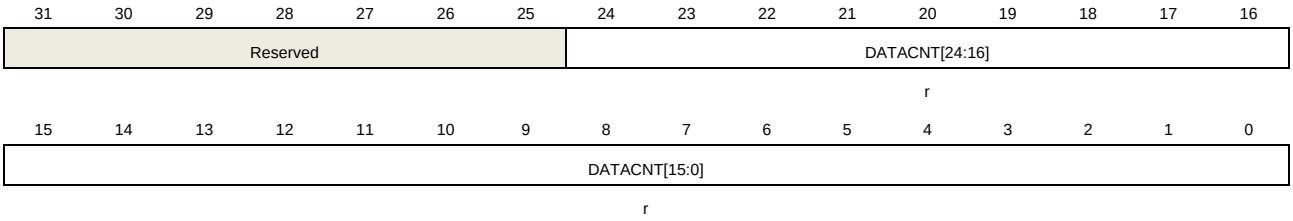
## 36.7.10. Data counter register (SDIO_DATACNT)

Address offset: 0x30

Reset value: 0x0000 0000

This register is read only. When the DSM from Idle to WaitR or WaitS, it loads value from data length register (SDIO_DATALEN). It decrements with the data transferred, when it reaches 0, the flag DTEND is set.

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24:0 | DATACNT[24:0] | Data count value<br>Read-only bits field. When these bits are read, the number of remaining data bytes to be transferred is returned. |

**Note:** This register should be read only after transmission is completed or held. When reading data after an error event, the read count value may be different from the actual number of data

bytes transferred.

### 36.7.11. Status register (SDIO_STAT)

Address offset: 0x34
Reset value: 0x0000 0000

This register is read only. The following descripts the types of flag:

The flags of bit [28:21, 11:0] can only be cleared by writing 1 to the corresponding bit in interrupt clear register (SDIO_INTC).

The flags of bit [20:12] are changing depend on the hardware logic.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | IDMAEND | IDMAERR | CLKSTOP | VSEND | ACKTO | ACKFAIL | SDIOINT | DAT0BSYEND | DAT0BSY | RFE | TFE | RFF | TFF |
| | | | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFH | TFH | DATSTA | CMDSTA | DATABOR | DTBLKEND | DATHOLD | DTEND | CMDSEND | CMDRECV | RXORE | TXURE | DTTMOUT | CMDTMOUT | DTCRCERR | CCRCERR |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | IDMAEND | IDMA transfer end |
| 27 | IDMAERR | IDMA transfer error |
| 26 | CLKSTOP | SDIO_CLK stopped in Voltage Switch procedure |
| 25 | VSEND | Voltage Switch critical timing section end |
| 24 | ACKTO | Boot acknowledgment timeout |
| 23 | ACKFAIL | Boot acknowledgment received and check fail |
| 22 | SDIOINT | SD I/O interrupt received |
| 21 | DAT0BSYEND | End of CMD response, card DAT0 line signal changed from busy to ready. |
| 20 | DAT0BSY | End of CMD response and a second time 2 clock cycles after the CMD response, card signals busy on DAT0 line.<br>This is a hardware status flag only, does not generate an interrupt. |
| 19 | RFE | Receive FIFO is empty<br>This is a hardware status flag only, does not generate an interrupt. |
| 18 | TFE | Transmit FIFO is empty<br>This bit is cleared when one FIFO location becomes full. |
| 17 | RFF | Receive FIFO is full |

| | | |
|---|---|---|
| | | This bit is cleared when one FIFO location becomes empty. |
| 16 | TFF | Transmit FIFO is full<br>This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty. |
| 15 | RFH | Receive FIFO is half full: at least half the number of words can be read in the FIFO<br>This bit is cleared when the FIFO becomes (half + 1) empty. |
| 14 | TFH | Transmit FIFO is half empty: at least half the number of words can be written into the FIFO.<br>This bit is cleared when the FIFO becomes (half + 1) full. |
| 13 | DATSTA | Data path active state<br>This is a hardware status flag only, does not generate an interrupt. |
| 12 | CMDSTA | Command path active state<br>This is a hardware status flag only, does not generate an interrupt. |
| 11 | DATABOR | Data transfer aborted by CMD12 |
| 10 | DTBLKEND | Data block sent/received (CRC check passed) |
| 9 | DATHOLD | Data transfer hold |
| 8 | DTEND | Data end (data counter, SDIO_DATACNT is zero) |
| 7 | CMDSEND | Command sent (no response required) |
| 6 | CMDRECV | Command response received (CRC check passed) |
| 5 | RXORE | Received FIFO overrun error occurs |
| 4 | TXURE | Transmit FIFO underrun error occurs |
| 3 | DTTMOUT | Data timeout<br>The data timeout period depends on the SDIO_DATATO register. |
| 2 | CMDTMOUT | Command response timeout<br>The command timeout period has a fixed value of 64 SDIO_CLK clock periods. |
| 1 | DTCRCERR | Data block sent/received (CRC check failed) |
| 0 | CCRCERR | Command response received (CRC check failed) |

**Note:** If using IDMA mode, the FIFO Interrupt cannot be enabled.

### 36.7.12. Interrupt clear register (SDIO_INTC)

Address offset: 0x38
Reset value: 0x0000 0000

This register is write only. Writing 1 to the bit can clear the corresponding bit in the SDIO_STAT register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | IDMAENDC | IDMAERRC | CLKSTOPC | VSENDC | ACKTOC | ACKFAILC | SDIOINTC | DAT0BSYENDC | Reserved | | | | | |
| | | | w | w | w | w | w | w | w | w | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | DATABORC | DTBLKENDC | DATHOLDC | DTENDC | CMDSENDC | CMDRECVC | RXOREC | TXUREC | DTTMOUTC | CMDTMOUTC | DTCRCERRC | CCRCERRC |
| | | | | w | w | w | w | w | w | w | w | w | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | IDMAENDC | IDMAEND flag clear bit<br>Write 1 to this bit to clear the flag. |
| 27 | IDMAERRC | IDMAERR flag clear bit<br>Write 1 to this bit to clear the flag. |
| 26 | CLKSTOPC | CLKSTOP flag clear bit<br>Write 1 to this bit to clear the flag. |
| 25 | VSENDC | VSEND flag clear bit<br>Write 1 to this bit to clear the flag. |
| 24 | ACKTOC | ACKTO flag clear bit<br>Write 1 to this bit to clear the flag. |
| 23 | ACKFAILC | ACKFAIL flag clear bit<br>Write 1 to this bit to clear the flag. |
| 22 | SDIOINTC | SDIOINT flag clear bit<br>Write 1 to this bit to clear the flag. |
| 21 | DAT0BSYENDC | DAT0BSYEND flag clear bit<br>Write 1 to this bit to clear the flag. |
| 20:12 | Reserved | Must be kept at reset value. |
| 11 | DATABORC | DATABOR flag clear bit<br>Write 1 to this bit to clear the flag. |
| 10 | DTBLKENDC | DTBLKEND flag clear bit<br>Write 1 to this bit to clear the flag. |
| 9 | DATHOLDC | DATHOLD flag clear bit<br>Write 1 to this bit to clear the flag. |
| 8 | DTENDC | DTEND flag clear bit<br>Write 1 to this bit to clear the flag. |
| 7 | CMDSENDC | CMDSEND flag clear bit |

| | | |
|---|---|---|
| | | Write 1 to this bit to clear the flag. |
| 6 | CMDRECVC | CMDRECV flag clear bit<br>Write 1 to this bit to clear the flag. |
| 5 | RXOREC | RXORE flag clear bit<br>Write 1 to this bit to clear the flag. |
| 4 | TXUREC | TXURE flag clear bit<br>Write 1 to this bit to clear the flag. |
| 3 | DTTMOUTC | DTTMOUT flag clear bit<br>Write 1 to this bit to clear the flag. |
| 2 | CMDTMOUTC | CMDTMOUT flag clear bit<br>Write 1 to this bit to clear the flag. |
| 1 | DTCRCERRC | DTCRCERR flag clear bit<br>Write 1 to this bit to clear the flag. |
| 0 | CCRCERRC | CCRCERR flag clear bit<br>Write 1 to this bit to clear the flag. |

### 36.7.13. Interrupt enable register (SDIO_INTEN)

Address offset: 0x3C

Reset value: 0x0000 0000

This register enables the corresponding interrupt in the SDIO_STAT register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | IDMAENDIE | IDMAERRIE | CLKSTOPIE | VSENDIE | ACKTOIE | ACKFAILIE | SDIOINTIE | DAT0BSYENDIE | Reserved | | TFEIE | RFFIE | Reserved |
| | | | rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFHIE | TFHIE | Reserved | | DATABORIE | DTBLKENDIE | DATHOLDIE | DTENDIE | CMDSENDIE | CMDRECVIE | RXOREIE | TXUREIE | DTTMOUTIE | CMDTMOUTIE | DTCRCERRIE | CCRCERRIE |
| rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | IDMAENDIE | IDMA transfer end interrupt enable |
| 27 | IDMAERRIE | IDMA transfer error interrupt enable |
| 26 | CLKSTOPIE | Voltage Switch clock stopped interrupt enable |
| 25 | VSENDIE | Voltage Switch critical timing section end interrupt enable |
| 24 | ACKTOIE | Boot acknowledgment timeout enable |

| | | |
|---|---|---|
| 23 | ACKFAILIE | Boot acknowledgment received and check fail interrupt enable |
| 22 | SDIOINTIE | SD I/O interrupt received interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 21 | DAT0BSYENDIE | DAT0 line signal changed from busy to ready interrupt enable. |
| 20:19 | Reserved | Must be kept at reset value. |
| 18 | TFEIE | Transmit FIFO empty interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 17 | RFFIE | Receive FIFO full interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 16 | Reserved | Must be kept at reset value. |
| 15 | RFHIE | Receive FIFO half full interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 14 | TFHIE | Transmit FIFO half empty interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 13:12 | Reserved | Must be kept at reset value. |
| 11 | DATABORIE | Data transfer abort interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 10 | DTBLKENDIE | Data block end interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 9 | DATHOLDIE | Data hold interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 8 | DTENDIE | Data end interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 7 | CMDSENDIE | Command sent interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 6 | CMDRECVIE | Command response received interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 5 | RXOREIE | Received FIFO overrun error interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 4 | TXUREIE | Transmit FIFO underrun error interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 3 | DTTMOUTIE | Data timeout interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 2 | CMDTMOUTIE | Command response timeout interrupt enable |

| | | |
|---|---|---|
| | | Write 1 to this bit to enable the interrupt. |
| 1 | DTCRCERRIE | Data CRC fail interrupt enable<br>Write 1 to this bit to enable the interrupt. |
| 0 | CCRCERRIE | Command response CRC fail interrupt enable<br>Write 1 to this bit to enable the interrupt. |

## 36.7.14. ACK timeout register (SDIO_ACKTO)

Address offset: 0x40
Reset value: 0x0000 0000

When DSM enters the WaitACK state, the SDIO_ACKTO register value will loaded into a counter. It is used to answer whether the acknowledgment timeout occurs. If the counter reaches 0 and the DSM is in this state, the ACK timeout status flag is set to 1.

The data transfer must be written to the SDIO_ACKTO register before being written to the SDIO_DATACTL register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | ACKTO[24:16] | | | | | | | | |
| | | | | | | | rw | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ACKTO[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24:0 | ACKTO[24:0] | Boot ACK timeout period.<br>These bits can only be configured by firmware when CSMEN = 0. |

## 36.7.15. FIFO data register (SDIO_FIFO)

Address offset: 0x80
Reset value: 0x0000 0000

This register occupies 16 entries of 32-bit words, the address offset is from 0x80 to 0xBC.

This register has to be accessed by word(32-bit).

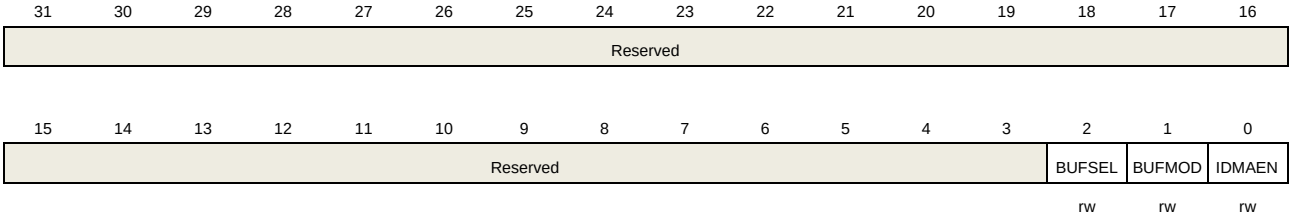
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIFODT[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FIFODT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FIFODT[31:0] | Receive FIFO data or transmit FIFO data<br>These bits are the data of receive FIFO or transmit FIFO. Write to or read from this register is write data to FIFO or read data from FIFO.<br>This register can only be read or written by firmware when DATSTA = 1. |

## 36.7.16. IDMA control register (SDIO_IDMACTL)

Address offset: 0x50
Reset value: 0x0000 0000

The transmit and receive FIFO can be written or read as a 32-bit wide register. The FIFO have 32 consecutive addresses as 32 entries. This allows the CPU to use its store/load multiple operator commands to write/read FIFO.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | BUFSEL | IDMA double buffer selection bit<br>This bit can only be configured by firmware when DATSTA = 0.<br>When IDMA is enabled, this bit is toggled by hardware.<br>0: If IDMA is enabled, uses buffer0, and firmware write access to IDMAADDR0 is not allow.<br>1: If IDMA is enabled, uses buffer1, and firmware write access to IDMAADDR1 is not allow. |
| 1 | BUFMOD | Double buffer selection bit<br>This bit can only be configured by firmware when DATSTA = 0.<br>0: Single buffer mode.<br>1: Double buffer mode. |
| 0 | IDMAEN | FIFO internal DMA enable bit<br>This bit can only be configured by firmware when DATSTA = 0.<br>0: IDMA disable.<br>1: IDMA enable. |

## 36.7.17. IDMA buffer size register (SDIO_IDMASIZE)

Address offset: 0x54
Reset value: 0x0000 0000

This register contains the buffers size when in double buffer configuration.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | IDMASIZE[7：0] | | | | | | | | Reserved | | | | |

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12:5 | IDMASIZE[7：0] | Number of bytes per buffer.<br>IDMA buffer size = IDMASIZE[7：0] * 8 words.<br>0x00: buffer size = 0 words<br>0x01: buffer size = 8 words<br>0x02: buffer size = 16 words<br>……<br>0xFF: buffer size = 2040 words |
| 4:0 | Reserved | Must be kept at reset value. |

## 36.7.18. IDMA buffer 0 base address register (SDIO_IDMAADDR0)

Address offset: 0x58

Reset value: 0x0000 0000

This register contains the base address of memory buffer in single buffer configuration and the base address of buffer 0 in double buffer configuration.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDMAADDR0[31:16] | | | | | | | | | | | | | | | |

rw

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDMAADDR0[15:0] | | | | | | | | | | | | | | | |

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | IDMAADDR0[31:0] | The address is a multiple of 4. The IDMAADDR0[1:0] are always 0 and read only. This register can be written by firmware when DATSTA = 0, and can dynamically be written by firmware when DATSTA and BUFSEL all set to 1 . |

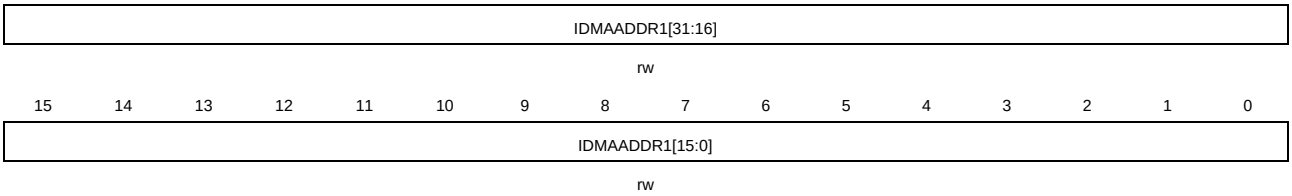
## 36.7.19. IDMA buffer 1 base address register (SDIO_IDMAADDR1)

Address offset: 0x5C

Reset value: 0x0000 0000

This register contains t the base address of buffer 1 in double buffer configuration.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | IDMAADDR1[31:0] | The address is a multiple of 4. The IDMAADDR1[1:0] are always 0 and read only. This register can be written by firmware when DATSTA = 0, and can dynamically be written by firmware when DATSTA = 1 and BUFSEL = 0. |

# 37. Management data input / output (MDIO) (For GD32H737_757_759)

## 37.1. Overview

The MDIO interface can receive complete MDIO frames. As long as the data is written to the register before receiving the turnaround bits (TA) of the read or post read increment address frame, the MDIO interface can transmit complete MDIO frames. Interrupts are generated at the end of every complete frame, which can be used or provided at correct time. Interrupts can also be generated after every valid PHYADR and DEVADD, which allows more complex controls within frames.

## 37.2. Characteristics

- Support slave mode with a frequency up to 4MHz.
- Support CFP / CFP2 MSA Management Interface Specification.
- Support various kinds of interrupts.
- Physical address can be setted:
  - By software;
  - By hardware pin.

## 37.3. Pins and internal signals

*Figure 37-1. CFP Management Interface Architecture* shows the MDIO block diagram. *Table 37-1. MDIO pins definition* gives the MDIO internal signals and pins description.

**Table 37-1. MDIO pins definition**

| Name | Signal type |
|---|---|
| MDIO_MDIO | Input / Output, Digital signal |
| MDIO_MDC | Input, Digital clock signal |
| MDIO_Ax(x=0…4) | Input, Digital address signals |

## 37.4. Function overview

**Figure 37-1. CFP Management Interface Architecture**

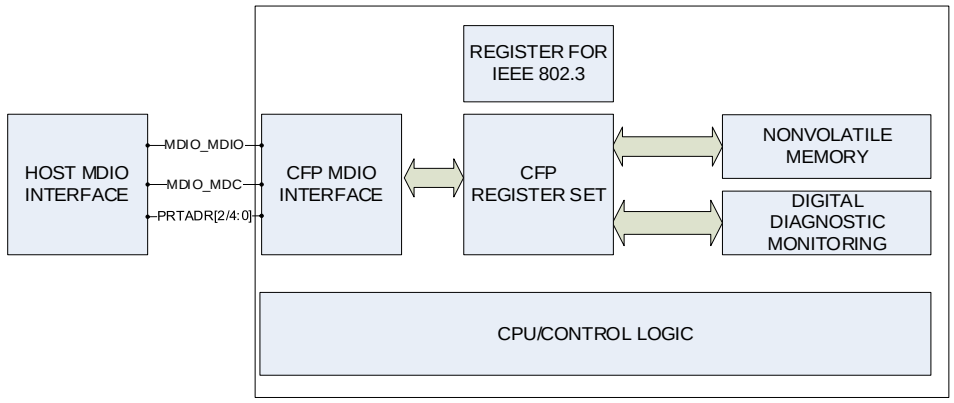
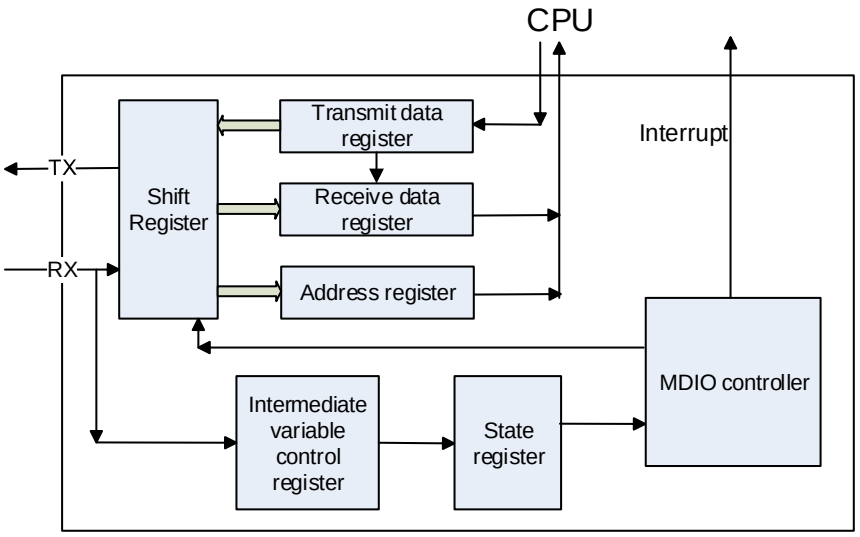


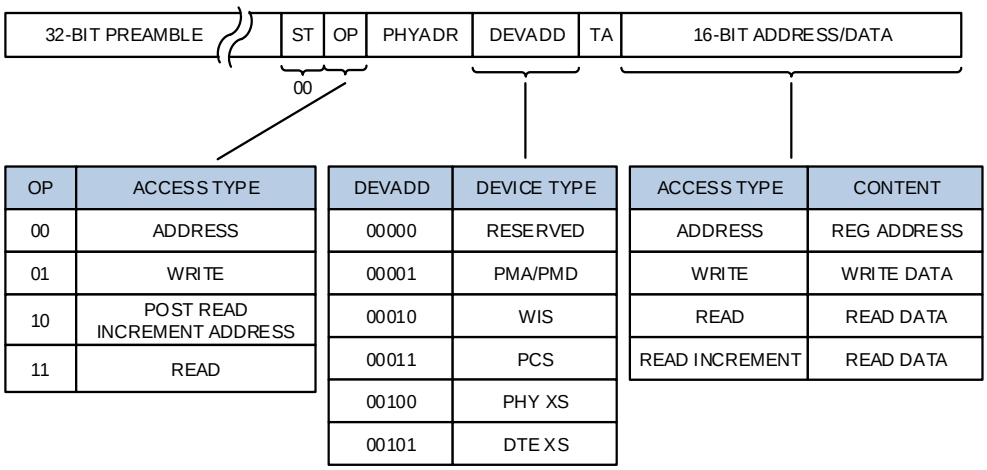

**Figure 37-2. MDIO Architecture**




### 37.4.1. Frame structure

CFP MDIO interface uses the communication data frame structure defined in IEEE 802.3 Clause 45. Each frame can be either an address frame or a data frame. The total bit length of each frame is 64, consisting of 32 bits preamble, and the frame command body.The command body consists of 6 parts illustrated in ***Figure 37-3. MDIO Frame Structure***.

**Figure 37-3. MDIO Frame Structure**




**Note:**

1. ST = Start bits (2 bits)
2. OP = Operation code (2 bits)
3. PHYADR = Physical port Address (5 bits)
4. DEVADD = MDIO Device Address (or called Device Type, 5 bits)
5. TA = Turnaround bits (2 bits)
6. 16-BIT ADDRESS/DATA is the payload

**Table 37-2. Frame Details for Different Frame Types (1)**

| Frame | Idle | Management Frame Fields | | | | | | | Idle |
|---|---|---|---|---|---|---|---|---|---|
| | | PRE | ST | OP | PHYADR | DEVADD | TA | Address/Data | |
| Write Address | Z | 1…1 | 00 | 00 | aaaaa | aaaaa | 10 | aaaaaaaaaaaaaaaa | Z |
| Write Data | Z | 1…1 | 00 | 01 | aaaaa | aaaaa | 10 | dddddddddddddddd | Z |
| Read Data | Z | 1…1 | 00 | 11 | aaaaa | aaaaa | z0 | dddddddddddddddd | Z |
| Post Read Increment Address Frame | Z | 1…1 | 00 | 10 | aaaaa | aaaaa | z0 | dddddddddddddddd | Z |

(1). During the idle condition, the MDIO clock (MDC) and MDIO are not actively driven. For a read or a post read increment address frame, the TA bits is 1.5 bits, and the ahead 0.5 bit cycle of TA is driven by the host, the other one bit cycle of TA and during the 16-bit data time, MDIO is driven by the MDIO manageable device (slave), then another 0.5 bit cycle of TA is followed, MDIO is driven by the host. At all other times, the station management entity (host) drives the MDIO.

**Idle Condition (Idle)**

The idle condition for the MDIO is a high impedance state.

**Preamble (PRE)**

At the beginning of each transaction, the station management entity (host) sends a sequence of at least 32 contiguous bits of bit ‘1’ at a time to the MDIO, with 32 corresponding clock cycles on the MDIO clock (MDC), to establish the start of a frame.

**Start of Frame (ST)**

After PRE, the ST (consisting of two zero bits) indicates the start of the frame information.

**Operation Code (OP)**

The OP specifies the action to take, as described in ***Table 37-3. Operation Code***.

**Table 37-3. Operation Code**

| OP | Descriptions |
|---|---|
| 00 | Set the address for a subsequent write or read frame. |
| 01 | Write to the previously set address. |
| 11 | Read from the previously set address. |
| 10 | Read from the previously set address, and then increment the address. User code must increment the address. |

**Physical Address (PHYADR)**

The physical address is five bits, allowing 32 unique addresses. PHYADR is set either by five pins or by the software.

**Device Address (DEVADD)**

DEVADD is five bits and selects the device type. In the CFP standard, only MDIO Device Address 1 is supported.

**Turnaround (TA)**

The MDIO changes from the station management entity (host) to MDIO manageable device (slave) during the TA time.

**Address / Data**

The address / data field is 16 bits.

### 37.4.2. Typical Usage Sequence

Most of the MDIO interface is implemented in the hardware, requiring correct software operation. The following is the usage flow:

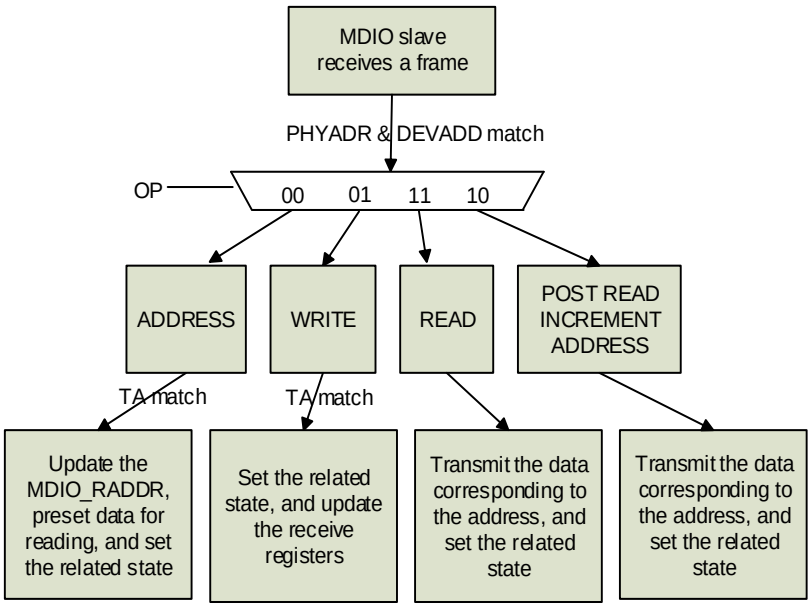
1. Reset the MDIO and configure the GPIO module to make selected PADs to alternate function.
2. Set the frame parameters by writing the MDIO_CTL, MDIO_CFG and MDIO_TO registers as you need.
3. Set the interrupts by writing the MDIO_INTEN register and the required system interrupt settings.
4. For a write frame, the write address and write data frames can be received in the MDIO_RADDR and MDIO_RDATA registers after receiving complete frame. For a read data frame or post read increment address frame, data must be placed in the MDIO_TDATA register in advance of the frame so that the data can be automatically inserted for the frame.

In this process, no software intervention is required. The frame progress can be monitored with the MDIO_RFRM register when frame field match or unmatch event occur during each frame or at the completion of each frame. Due to bit [9:0] of MDIO_STAT register is automatically cleared, do not use MDIO_STAT register to monitor frame progress, and the MDIO_STAT register can be lost if read at an inappropriate time. Read the MDIO_STAT register only once per frame. To monitor frame progress, read the status in MDIO_STAT register according to interrupts or by polling the related bit 8 (interrupt num 40) in SETPEND1 register (in NVIC registers). MDIO interrupts must have the highest priority of all peripherals, otherwise they are easy to lose.

**Notes**:

1. The system clock frequency should be more than 3 times that of the MDC clock.
2. Be sure to send data to MDIO_TDATA register before TA.
3. Please configure the software reset before use if timeout or other error occurs.

**Figure 37-4. MDIO slave communication process**




After receiving the address frame sent by the host, no matter whether the next frame is write or read, the data can be preset to the transfer data register in advance, thus if the next frame is read, MDIO will send the preset data. If the address frame is received but no read operation is performed following, the new data can be overwritten to the original transfer data register when the next address frame is accepted.

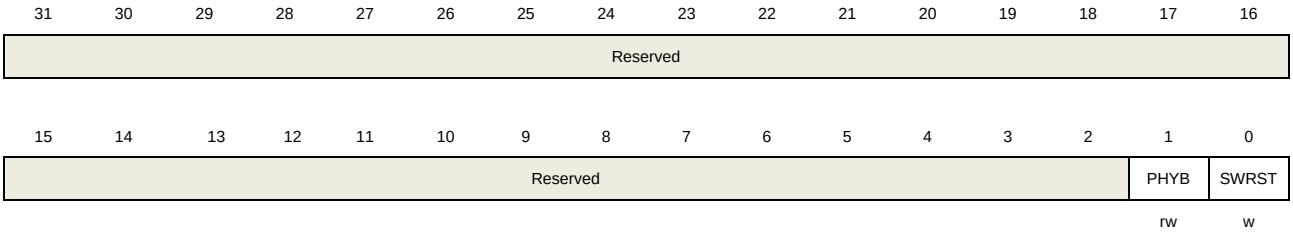
## 37.5. Register definition

MDIO base address: 0x4000 D800

### 37.5.1. Control register (MDIO_CTL)

Address offset: 0x00
Reset value: 0x0000 XXXX

This register has to be accessed by word (32-bit).

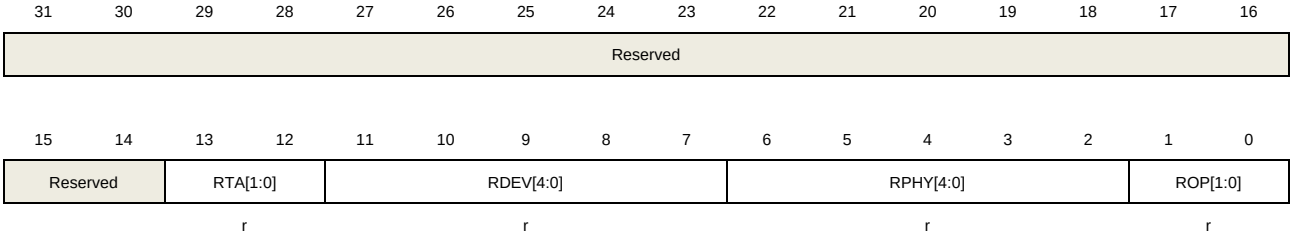



| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | PHYB | MDIO PHY bit length<br>0: MDIO PHY uses 5 bits<br>1: MDIO PHY uses 3 bits. Unused PHY bits are ignored |
| 0 | SWRST | Write 1 to reset MDIO block. Registers will not be reset. Hardware immediately clears this bit. |

### 37.5.2. Received frame information register (MDIO_RFRM)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13:12 | RTA[1:0] | Received frame field TA (only accept TA bits of a write data frame or a write address frame) |

| | | |
|---|---|---|
| 11:7 | RDEV[4:0] | Received frame field DEVADD |
| 6:2 | RPHY[4:0] | Received frame field PHYADR |
| 1:0 | ROP[1:0] | Received frame field OP<br>00: write address frame<br>01: write data frame<br>10: post read increment address frame<br>11: read data frame |

### 37.5.3. Received data register (MDIO_RDATA)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | RDATA[15:0] | Received frame field DATA |

### 37.5.4. Received address register (MDIO_RADDR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RADDR[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

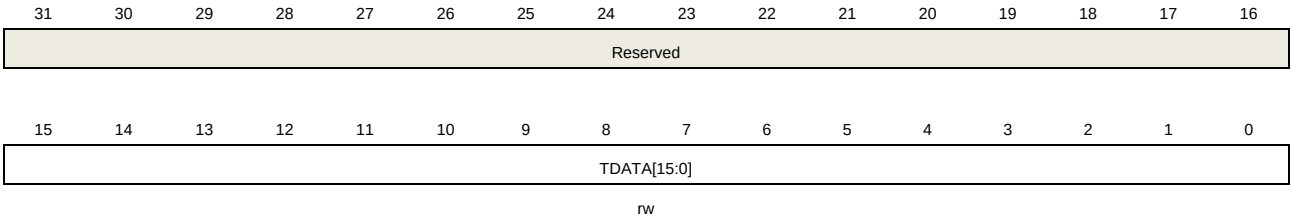
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | RADDR[15:0] | Received frame field ADDRESS |

### 37.5.5. Transfer data register (MDIO_TDATA)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



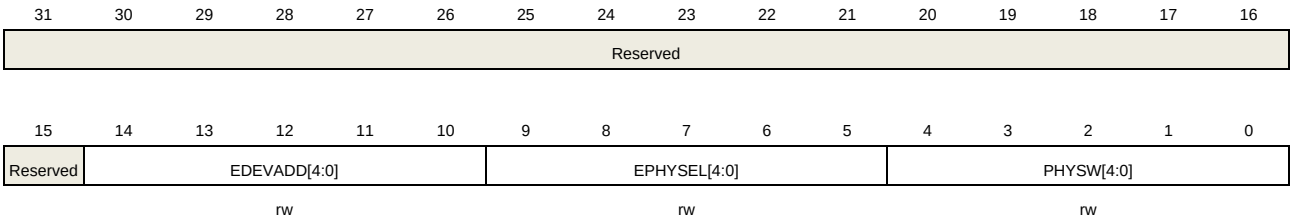

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | TDATA[15:0] | Data that is transmitted by the next read or post read increment address frame. Before a read data frame, the master sends a write address frame to specify which data is to be read. After this write address frame, the user software must place this requested data into MDIO_TDATA before it is required by the read frame. The time available is at latest 3 MDIO clock cycles before TA. |

### 37.5.6. Configuration register (MDIO_CFG)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14:10 | EDEVADD[4:0] | Expected DEVADD. Normally 01. |
| 9:5 | EPHYSEL[4:0] | Selects expected PHYADR. For each bit x in the 5-bit EPHYSEL:<br>0: sets expected PHYADR.x = PHYPIN[4:0].x<br>1: sets expected PHYADR.x = PHYSW[4:0].x |
| 4:0 | PHYSW[4:0] | Software provided PHYADR. Chosen according to corresponding EPHYSEL[4:0] bits. |

### 37.5.7. Status register (MDIO_STAT)

Address offset: 0x18

Reset value: 0x0000 1000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | RBNE | Reserved | OVR | UDR | TO | TANM | PHYNM | PHYM | DEVNM | DEVM | RDFRM | RDINCFRM | ADDRFRM | WRFRM |
| | | r | | r | r | rc | rc | rc | rc | rc | rc | rc | rc | rc | rc |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | RBNE | Read data buffer not empty flag. Data is received and ready to be read. Cleared by reading the MDIO_RDATA register. |
| 12 | Reserved | Must be kept at reset value. |
| 11 | OVR | Receive overrun flag. Set by hardware when the data in the receive shift register is ready to be transferred into the MDIO_RDATA register while the RBNE bit is set. Cleared by reading the MDIO_RDATA register. |
| 10 | UDR | Transmit underrun flag. Set by hardware if the data has not been written to MDIO_TDATA register 3 MDIO clock cycles before TA of a read / post read increment address frame when the MDIO_TDATA is empty. Cleared by a write to the MDIO_TDATA register. |
| 9 | TO | Timeout flag. If a new bit of a frame is not received or a bit of the prepared data is not transmitted before reaching the configured timeout value, this bit will be set by hardware. Cleared by reading the MDIO_STAT register. |
| 8 | TANM | Set at end bit of TA of a write data frame or a write address frame if the received TA nonmatches expected ‘10’. Cleared by reading the MDIO_STAT register. |
| 7 | PHYNM | Set at end bit of PHYADR if PHYADR nonmatches. Cleared by reading the MDIO_STAT register. |
| 6 | PHYM | Set at end bit of PHYADR if PHYADR matches. Cleared by reading the MDIO_STAT register. |
| 5 | DEVNM | Set at end bit of DEVADD if DEVADD nonmatches. Cleared by reading the MDIO_STAT register. |
| 4 | DEVM | Set at end bit of DEVADD if DEVADD matches. Cleared by reading the MDIO_STAT register. |

| | | |
|---|---|---|
| 3 | RDFRM | Set at end bit (after frame field DATA transmission) of a read data frame if DEVADD and PHYADR both match. Cleared by reading the MDIO_STAT register. |
| 2 | RDINCFRM | Set at end bit (after frame field DATA transmission) of a post read increment address frame if DEVADD and PHYADR both match. Cleared by reading MDIO_STAT register. |
| 1 | ADDRFRM | Set at end bit of a write address frame if DEVADD and PHYADR both match. Cleared by reading the MDIO_STAT register. |
| 0 | WRFRM | Set at end bit of a write data frame if DEVADD and PHYADR both match. Cleared by reading the MDIO_STAT register. |

### 37.5.8. Interrupt enable register (MDIO_INTEN)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 120 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | RBNEIE | Reserved | OVRIE | UDRIE | TOIE | TANMIE | PHYNMIE | PHYMIE | DEVNMIE | DEVMIE | RDFRMIE | RDINCFRMIE | ADDRFRMIE | WRFRMIE |
| | | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

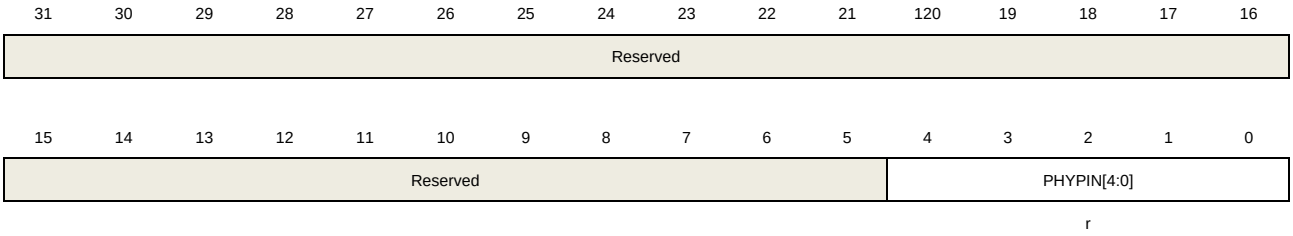
| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | RBNEIE | If set, interrupt is requested when RBDE bit in MDIO_STAT register becomes active. |
| 12 | Reserved | Must be kept at reset value. |
| 11 | OVRIE | If set, interrupt is requested when OVR bit in MDIO_STAT register becomes active. |
| 10 | UDRIE | If set, interrupt is requested when UDR bit in MDIO_STAT register becomes active. |
| 9 | TOIE | If set, interrupt is requested when TO bit in MDIO_STAT register becomes active. |
| 8 | TANMIE | If set, interrupt is requested when TANM bit in MDIO_STAT register becomes active. |
| 7 | PHYNMIE | If set, interrupt is requested when PHYNM bit in MDIO_STAT register becomes active. |
| 6 | PHYMIE | If set, interrupt is requested when PHYM bit in MDIO_STAT register becomes active. |
| 5 | DEVNMIE | If set, interrupt is requested when DEVNM bit in MDIO_STAT register becomes |

| | | |
|---|---|---|
| | | active. |
| 4 | DEVMIE | If set, interrupt is requested when DEVM bit in MDIO_STAT register becomes active. |
| 3 | RDFRMIE | If set, interrupt is requested when RDFRM bit in MDIO_STAT register becomes active. |
| 2 | RDINCFRMIE | If set, interrupt is requested when RDINCFRM bit in MDIO_STAT register becomes active. |
| 1 | ADDRFRMIE | If set, interrupt is requested when ADDRFRM bit in MDIO_STAT register becomes active. |
| 0 | WRFRMIE | If set, interrupt is requested when WRFRM bit in MDIO_STAT register becomes active. |

### 37.5.9. Pin value register (MDIO_PIN)

Address offset: 0x20
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

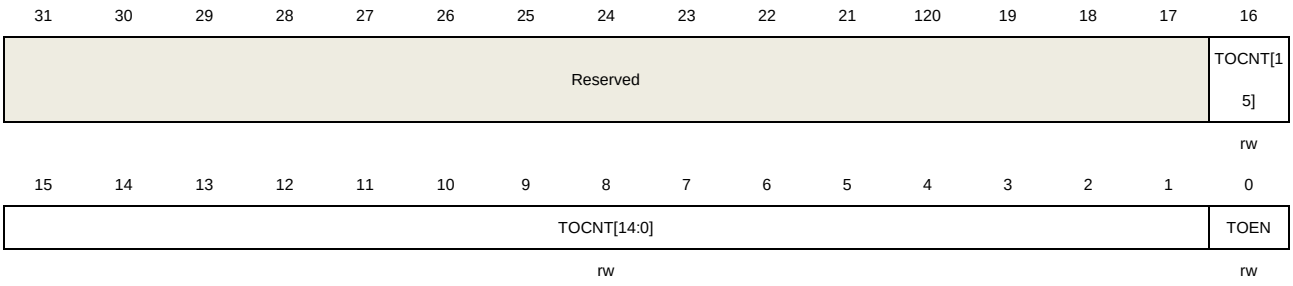



| Bits | Fields | Descriptions |
|---|---|---|
| 31:5 | Reserved | Must be kept at reset value. |
| 4:0 | PHYPIN[4:0] | Pin value read from hardware MDIO_Ax(x=0…4) pins |

### 37.5.10. Timeout register (MDIO_TO)

Address offset: 0x24
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16:1 | TOCNT[15:0] | MDIO time out = TOCNT[15:0]*PCLK1 |
| 0 | TOEN | Timeout enable<br>0: Timeout is disabled<br>1: Timeout is enabled |

# 38. External memory controller (EXMC)

## 38.1. Overview

The external memory controller EXMC, is used as a translator for CPU to access a variety of external memory, it automatically converts AXI memory access protocol into a specific memory access protocol defined in the configuration register, such as SRAM, ROM, NOR Flash, PSRAM, NAND Flash and SDRAM. Users could also tweak with the timing parameters in the configuration register to boost up memory access efficiency. EXMC access space is divided into multiple banks; each bank is assigned to access a specific memory type with flexible parameter configuration as defined in the controlling register.

## 38.2. Characteristics

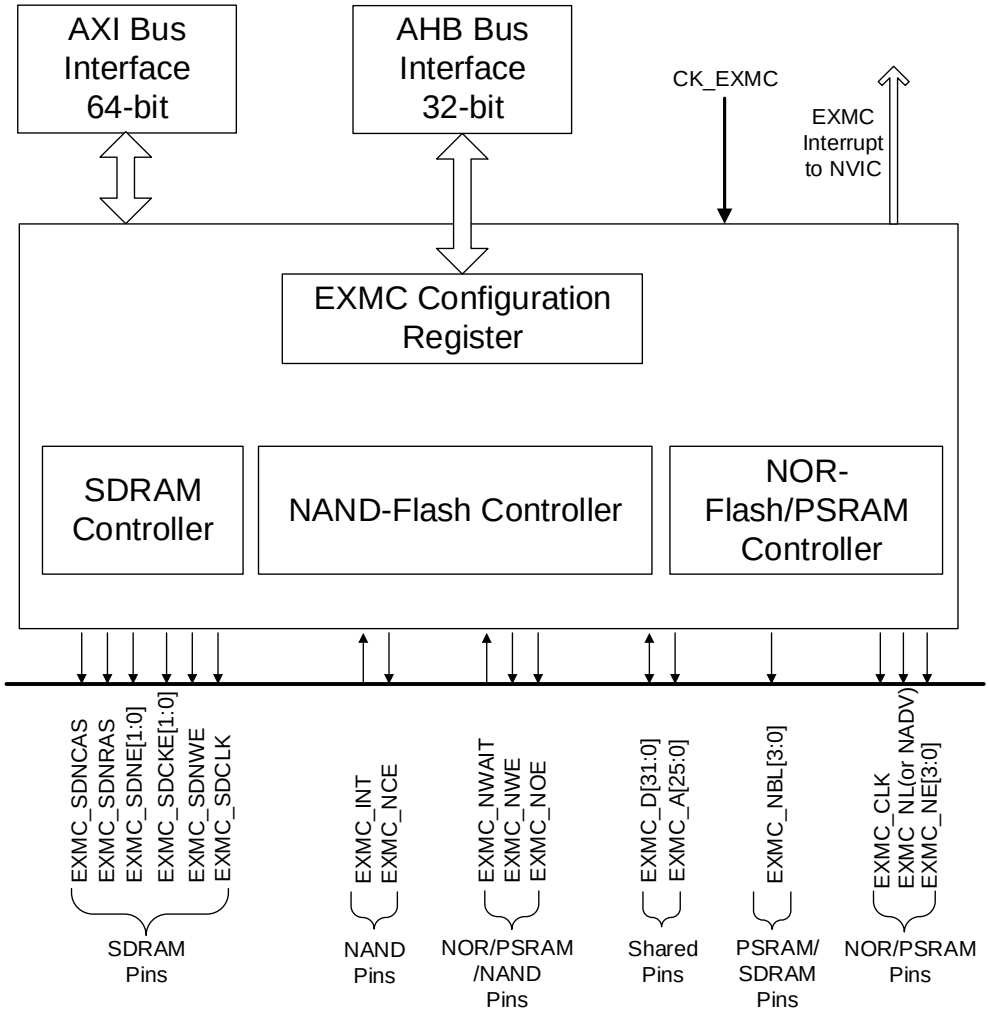
- Supported external memory:
  - SRAM
  - PSRAM
  - ROM
  - NOR Flash
  - 8-bit or 16-bit NAND Flash
  - Synchronous DRAM(SDRAM)
- Protocol translation between the AXI and the multitude of external memory protocol.
- Offering a variety of programmable timing parameters to meet user's specific needs.
- Each bank has its own chip-select signal which can be configured independently.
- Independent read / write timing configuration to a sub-set memory type.
- Embedded ECC hardware for NAND Flash access.
- 8,16, or 32 bits bus width.
- Address and data bus multiplexing mechanism for NOR Flash and PSRAM.
- Write enable and byte select are provided as needed.
- Automatic AXI transaction split when internal and external bus width is not compatible.

## 38.3. Function overview

### 38.3.1. Block diagram

EXMC is the combination of 7 modules: The AHB bus interface, AXI bus interface, EXMC configuration registers, NOR/PSRAM controller, NAND controller, SDRAM controller and external device interface. AHB clock (HCLK) is the reference clock, which is used to configure the EXMC registers.

**Figure 38-1. The EXMC block diagram**




**Note:** The EXMC_NE3, EXMC_A0-A4 and EXMC_A10-A15 are not supported on all devices for GD32H75E. Please refer to the datasheet for details.

### 38.3.2. Bus interface

AHB bus interface: The CPU configures the EXMC register through the AHB slave interface.

AXI bus interface: CPU and AXI bus master device access external memory through AXI bus slave interface.

The clock of NOR, NAND, SDRAM controller is the asynchronous CK_EXMC (refer to EXMCSEL bits in ***Clock configuration register 4 (RCU_CFG4)***).

### 38.3.3. AXI error

Accessing EXMC bank region x (x = 0,.., 3) which is not enabled will generate AXI slave error.

If NREN bit in EXMC_SNCTLx (x = 0,.., 3) register is set to 0, accessing EXMC NOR Flash memory area will generate AXI slave error.

For the write operation of SDRAM device (WPEN is set to 1) that has been write protected,

will generate AXI slave error.

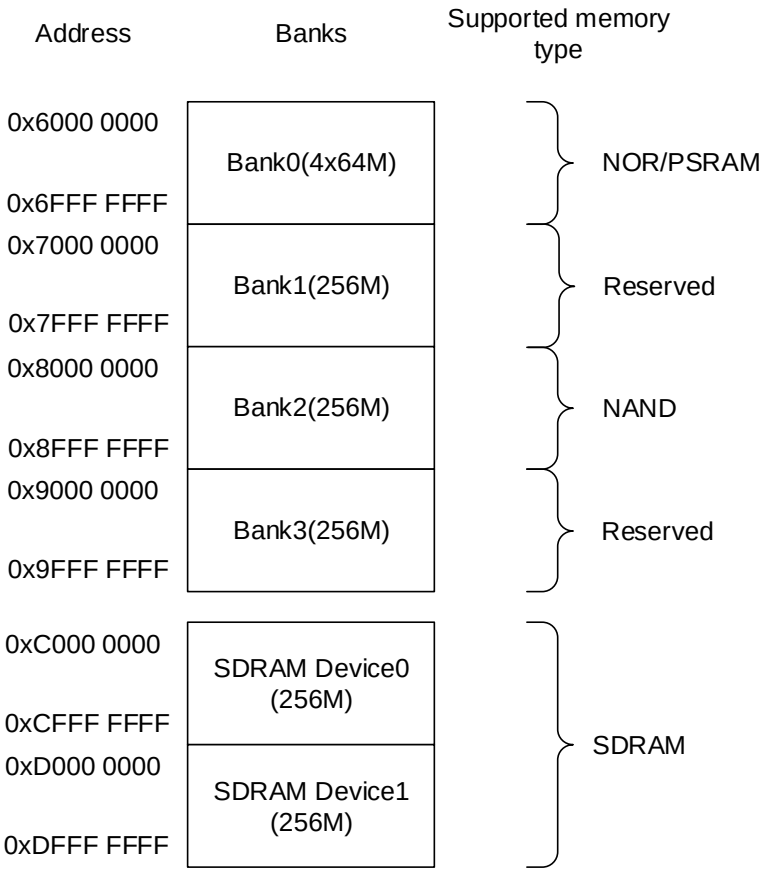
### 38.3.4. Basic regulation of EXMC access

EXMC is the conversion interface between AXI bus and external device protocol. Since the bit width of the AXI data bus is 64 bits, AXI transactions will split an access into several consecutive 8-bit, 16-bit or 32-bit accesses according to the size of the data. In the process of data transfer, AXI transaction data width and memory data width may not be the same. In order to ensure consistency of data transmission, EXMC's read / write accesses follow the following basic regulation.

- When the width of AXI transaction data width equals to the memory bus width, no conversion is applied.
- When the width of AXI transaction data width is greater than memory bus width, the AXI accesses will automatically split into several continuous memory accesses.
- When the width of AXI transaction data width is smaller than memory bus width, if the external memory devices have the byte selection function, such as SRAM, ROM, PSRAM, SDRAM, the application can access the corresponding byte through their byte lane EXMC_NBL[3:0]. Otherwise, write operation is prohibited, but read operation is allowed unconditionally. (See ***Table 38-17. Bank2 of EXMC support the memory and access mode***)

### 38.3.5. External device address mapping

**Figure 38-2. EXMC memory banks**




EXMC access space is divided into multiple banks. Each bank is 256 Mbytes. The first bank (Bank0) is further divided into four regions, and each region is 64 Mbytes. Bank2 is each divided into two spaces, the attribute memory space and the common memory space.

Each bank or region has a separate chip-select control signal, which can be configured independently.

Bank0 is used for NOR and PSRAM device access.

Bank2 are used to access NAND Flash exclusively.

SDRAM Device0 and Device1 are used for Synchronous DRAM (SDRAM) access.

The EXMC bank mapping can be modified through the BKREMAP[1:0] bis in the EXMC_SNCTL register. The EXMC bank mapping is shown in ***Table 38-1. EXMC bank mapping***.

**Table 38-1. EXMC bank mapping**

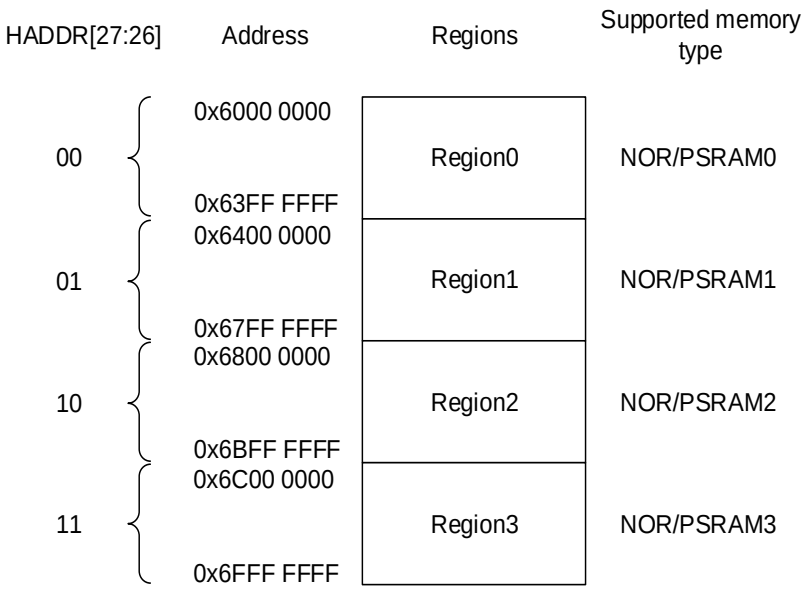
| Address | BKREMAP[1:0]=00 | BKREMAP[1:0]=01 |
|---|---|---|
| 0x6000 0000 – 0x6FFF FFFF | NOR/PSRAM bank | SDRAM Device 0 |

| Address | BKREMAP[1:0]=00 | BKREMAP[1:0]=01 |
|---|---|---|
| 0x7000 0000 – 0x7FFF FFFF | Reserved | |
| 0x8000 0000 – 0x8FFF FFFF | NAND bank | |
| 0x9000 0000 – 0x9FFF FFFF | Reserved | |
| 0xC000 0000 – 0xCFFF FFFF | SDRAM Device 0 | NOR/PSRAM bank |
| 0xD000 0000 – 0xDFFF FFFF | SDRAM Device 1 | |

### NOR/PSRAM address mapping

*Figure 38-3. Four regions of bank0 address mapping* reflects the address mapping of the four regions of bank0. Internal address lines HADDR[27:26] bit are used to select the four regions.

**Figure 38-3. Four regions of bank0 address mapping**




HADDR[25:0] is the byte address whereas the external memory may not be byte accessed, this will lead to address inconsistency. EXMC can adjust HADDR to accommodate the data width of the external memory according to the following rules.

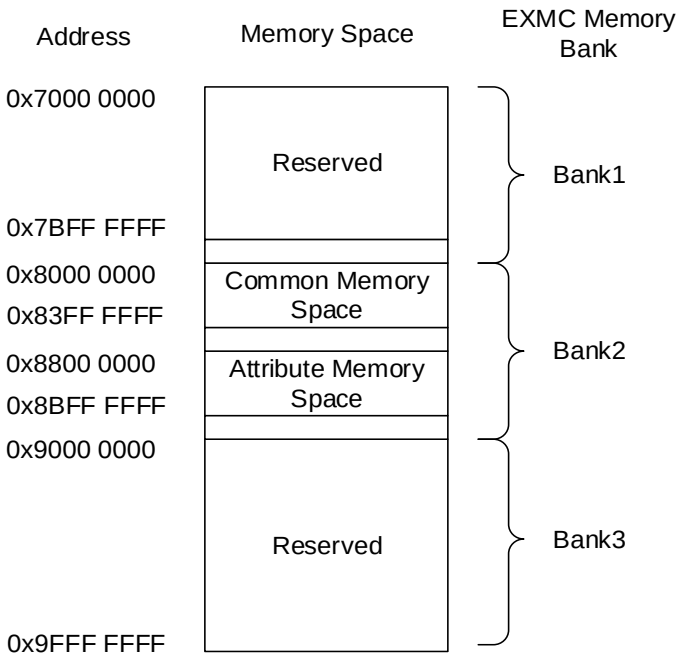
- When data bus width of the external memory is 8-bits. In this case the memory address is byte aligned. HADDR[25:0] is connected to EXMC_A[25:0] and then the EXMC_A[25:0] is connected to the external memory address lines.
- When data bus width of the external memory is 16-bits. In this case the memory address is half-word aligned. HADDR byte address must be converted into half-word aligned by connecting HADDR[25:1] with EXMC_A[24:0]. The EXMC_A[24:0] is connected to the external memory address lines.
- When data bus width of the external memory is 32-bits. In this case the memory address is word aligned. HADDR byte address must be converted into word aligned by connecting HADDR[25:2] with EXMC_A[23:0]. The EXMC_A[23:0] is connected to the

external memory address lines.

### NAND address mapping

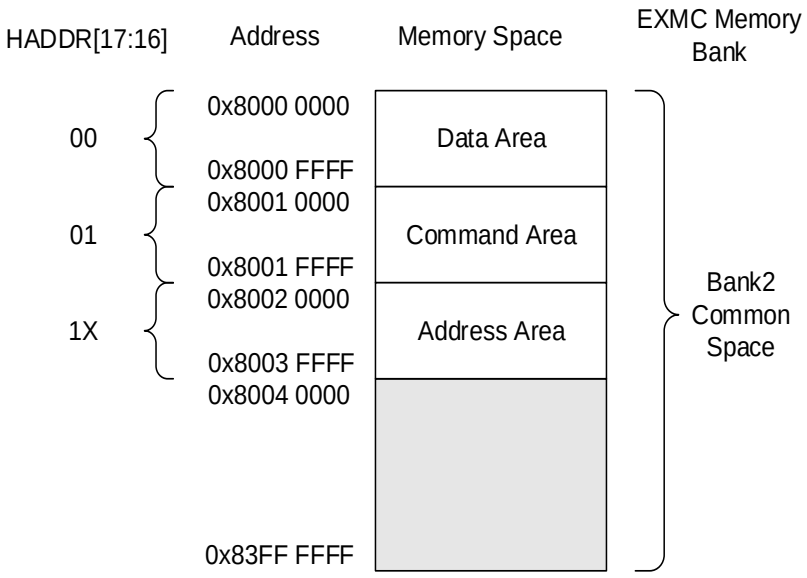
Bank2 is designed to access NAND Flash, bank1 and bank3 are reserved. Each bank is further divided into several memory spaces as shown in ***Figure 38-4. NAND address mapping***.

**Figure 38-4. NAND address mapping**




For NAND Flash, the common space and the attribute space are further-divided into three areas individually, the data area, the command area and the address area as shown in ***Figure 38-5. Diagram of bank2 common space***.

**Figure 38-5. Diagram of bank2 common space**




HADDR[17:16] bits are used to select one of the three areas.

- When HADDR[17:16] = 00, the data area is selected.
- When HADDR[17:16] = 01, the command area is selected.
- When HADDR[17:16] = 1X, the address area is selected.

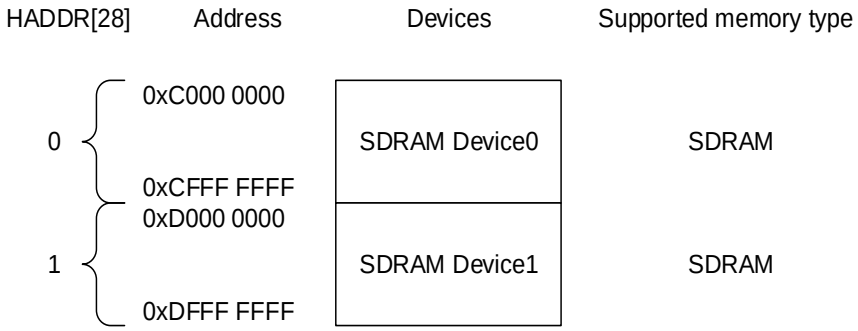
Application software uses these three areas to access NAND Flash, their definitions are as follows.

- Address area: This area is where the NAND Flash access address should be issued by software, the EXMC will pull the address latch enable (ALE) signal automatically in address transfer phase. ALE is mapped to EXMC_A[17].
- Command area: This area is where the NAND Flash access command should be issued by the software, the EXMC will pull the command latch enable (CLE) signal automatically in command transfer phase. CLE is mapped to EXMC_A[16].
- Data area: This area is where the NAND Flash read / write data should be accessed. When the EXMC is in data transfer mode, software should write the data to be transferred to the NAND Flash in this area. When the EXMC is in data reception mode, software should read the data from the NAND Flash by reading this area. Data access address is incremented automatically in consecutive mode, users do not need to be concerned with access address.

### SDRAM address mapping

The HADDR[28] bit is used to choose one of the two memory banks as shown in ***Figure 38-6. SDRAM address mapping***.

**Figure 38-6. SDRAM address mapping**




The ***Table 38-2. SDRAM mapping*** shows SDRAM address mapping of a 13-bit row and an 11-bit column configuration.

**Table 38-2. SDRAM mapping**

| Memory width | Internal bank | Row address | Column address | Maximum memory capacity |
|---|---|---|---|---|
| 8-bit | HADDR[25:24] | HADDR[23:11] | HADDR[10:0] | 64 Mbytes: 4 x 8K x 2K |
| 16-bit | HADDR[26:25] | HADDR[24:12] | HADDR[11:1] | 128 Mbytes: 4 x 8K x 2K x 2 |
| 32-bit | HADDR[27:26] | HADDR[25:13] | HADDR[12:2] | 256 Mbytes: 4 x 8K x 2K x 4 |

## 38.3.6. NOR/PSRAM controller

NOR/PSRAM memory controller controls bank0, which is designed to support NOR Flash, PSRAM, SRAM, ROM and honeycomb RAM external memory. EXMC has 4 independent chip-select signals for each of the 4 sub-banks within bank0, named NE[x] (x = 0, 1, 2, 3). Other signals for NOR/PSRAM access are shared. Each sub-bank has its own set of configuration register, and owns its corresponding unique register.

**Note:**

In asynchronous mode, all output signals of controller will change on the rise edge of internal CK_EXMC.

In synchronous mode, all output data of controller will change on the fall edge of extern memory device clock (EXMC_CLK).

### NOR/PSRAM memory device interface description

**Table 38-3. NOR flash interface signals description**

| EXMC Pin | Direction | Mode | Functional description |
|---|---|---|---|
| EXMC_CLK | Output | Sync | Clock signal for sync |
| Non-muxed | Output | Async/Sync | Address bus signal |

| EXMC Pin | Direction | Mode | Functional description |
|---|---|---|---|
| EXMC_A[25:0] | | | |
| Muxed EXMC_A[25:16] | | | |
| EXMC_D[15:0] | Input/output | Async/Sync (muxed) | Address/Data bus |
| | Input/output | Async/Sync (non-muxed) | Data bus |
| EXMC_NE[x] | Output | Async/Sync | Chip selection, x=0/1/2/3 |
| EXMC_NOE | Output | Async/Sync | Read enable |
| EXMC_NWE | Output | Async/Sync | Write enable |
| EXMC_NWAIT | Input | Async/Sync | Wait input signal |
| EXMC_NL(NADV) | Output | Async/Sync | Address valid |

**Table 38-4. PSRAM non-muxed signal description**

| EXMC Pin | Direction | Mode | Functional description |
|---|---|---|---|
| EXMC_CLK | Output | Sync | Clock signal for sync |
| EXMC_A[25:0] | Output | Async/Sync | Address Bus |
| EXMC_D[15:0] | Input/output | Async/Sync | Data Bus |
| EXMC_NE[x] | Output | Async/Sync | Chip selection, x=0/1/2/3 |
| EXMC_NOE | Output | Async/Sync | Read enable |
| EXMC_NWE | Output | Async/Sync | Write enable |
| EXMC_NWAIT | Input | Async/Sync | Wait input signal |
| EXMC_NL(NADV) | Output | Async/Sync | Latch enable (address valid enable, NADV) |
| EXMC_NBL[1] | Output | Async/Sync | Upper byte enable |
| EXMC_NBL[0] | Output | Async/Sync | Lower byte enable |

### Supported memory access mode

*Table 38-5. EXMC bank 0 supports all transactions* shows an example of the supported devices type, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM.

**Table 38-5. EXMC bank 0 supports all transactions**

| Memory | Access Mode | R/W | AXI Transaction Size | Memory Transaction Size | Comments |
|---|---|---|---|---|---|
| NOR Flash | Async | R | 8 | 16 | |
| | Async | W | 8 | 16 | Not allowed |
| | Async | R | 16 | 16 | |
| | Async | W | 16 | 16 | |
| | Async | R | 32 | 16 | Split into 2 EXMC accesses |

| Memory | Access Mode | R/W | AXI Transaction Size | Memory Transaction Size | Comments |
|---|---|---|---|---|---|
| | Async | W | 32 | 16 | Split into 2 EXMC accesses |
| | Async | R | 64 | 16 | Split into 4 EXMC accesses |
| | Async | W | 64 | 16 | Split into 4 EXMC accesses |
| | Sync | R | 8 | 16 | Not allowed |
| | Sync | R | 16 | 16 | |
| | Sync | R | 32 | 16 | |
| | Sync | R | 64 | 16 | |
| PSRAM | Async | R | 8 | 16 | |
| | Async | W | 8 | 16 | Use of byte lanes EXMC_NBL[1:0] |
| | Async | R | 16 | 16 | |
| | Async | W | 16 | 16 | |
| | Async | R | 32 | 16 | Split into 2 EXMC accesses |
| | Async | W | 32 | 16 | Split into 2 EXMC accesses |
| | Async | R | 64 | 16 | Split into 4 EXMC accesses |
| | Async | W | 64 | 16 | Split into 4 EXMC accesses |
| | Sync | R | 8 | 16 | Not allowed |
| | Sync | R | 16 | 16 | |
| | Sync | R | 32 | 16 | |
| | Sync | R | 64 | 16 | |
| | Sync | W | 8 | 16 | Use of byte lanes EXMC_NBL[1:0] |
| | Sync | W | 16 | 16 | |
| | Sync | W | 32 | 16 | |
| | Sync | W | 64 | 16 | |
| SRAM and ROM | Async | R | 8 | 16 | |
| | Async | R | 16 | 16 | |
| | Async | R | 32 | 16 | Split into 2 EXMC accesses |
| | Async | R | 64 | 16 | Split into 4 EXMC accesses |
| | Async | W | 8 | 16 | Use of byte lanes EXMC_NBL[1:0] |

| Memory | Access Mode | R/W | AXI Transaction Size | Memory Transaction Size | Comments |
|---|---|---|---|---|---|
| | Async | W | 16 | 16 | |
| | Async | W | 32 | 16 | Use of byte lanes EXMC_NBL[1:0] |
| | Async | W | 64 | 16 | Use of byte lanes EXMC_NBL[1:0] |

### NOR Flash/PSRAM controller timing

EXMC provides various programmable timing parameters and timing models for SRAM, ROM, PSRAM, NOR Flash and other external static memory.

**Table 38-6. NOR / PSRAM controller timing parameters**

| Parameter | Function | Access mode | Unit | Min | Max |
|---|---|---|---|---|---|
| CKDIV | Sync Clock divide ratio | Sync | CK_EXMC | 2 | 16 |
| DLAT | Data latency | Sync | EXMC_CLK | 2 | 17 |
| BUSLAT | Bus latency | Async/Sync read | CK_EXMC | 0 | 15 |
| DSET | Data setup time | Async | CK_EXMC | 1 | 255 |
| AHLD | Address hold time | Async(muxed) | CK_EXMC | 1 | 15 |
| ASET | Address setup time | Async | CK_EXMC | 0 | 15 |

**Table 38-7. EXMC timing models**

| Timing model | | Extend mode | Mode description | Write timing parameter | Read timing parameter |
|---|---|---|---|---|---|
| Async | Mode 1 | 0 | SRAM/PSRAM/CRAM | DSET<br>ASET | DSET<br>ASET |
| | Mode 2 | 0 | NOR Flash | DSET<br>ASET | DSET<br>ASET |
| | Mode A | 1 | SRAM/PSRAM/CRAM with EXMC_OE toggling on data phase | WDSET<br>WASET | DSET<br>ASET |
| | Mode B | 1 | NOR Flash | WDSET<br>WASET | DSET<br>ASET |
| | Mode C | 1 | NOR Flash with EXMC_OE toggling on data phase | WDSET<br>WASET | DSET<br>ASET |
| | Mode D | 1 | With address hold capability | WDSET<br>WAHLD<br>WASET | DSET<br>AHLD<br>ASET |
| | Mode AM | 0 | NOR Flash address/data mux | DSET<br>AHLD<br>ASET<br>BUSLAT | DSET<br>AHLD<br>ASET<br>BUSLAT |
| Sync | Mode E | 0 | NOR/PSRAM/CRAM | DLAT | DLAT |

| Timing model | | Extend mode | Mode description | Write timing parameter | Read timing parameter |
|---|---|---|---|---|---|
| | | | synchronous read PSRAM/CRAM synchronous write | CKDIV | CKDIV |
| | Mode SM | 0 | NOR Flash address/data mux | DLAT CKDIV | DLAT CKDIV |

As shown in ***Table 38-7. EXMC timing models***, EXMC NOR Flash / PSRAM controller provides a variety of timing model, users can modify those parameters listed in ***Table 38-6. NOR / PSRAM controller timing parameters*** to satisfy different external memory type and user's requirements. When extended mode is enabled via the EXMODEN bit in EXMC_SNCTLx register, different timing patterns for read and write access could be generated independently according to EXMC_SNTCFGx and EXMC_SNWTCFGx register's configuration.

EXMC_CLK can be configured through the consecutive clock (CCK) bit. If CCK is set to 0, when NOR flash synchronous access is performed, EXMC_CLK will be generated. If CCK is set to 1, EXMC_CLK will be generated unconditionally whether the NOR flash is accessed in synchronous or asynchronous mode.

### Asynchronous access timing diagram

Mode 1 - SRAM/CRAM

**Figure 38-7. Mode 1 read access**

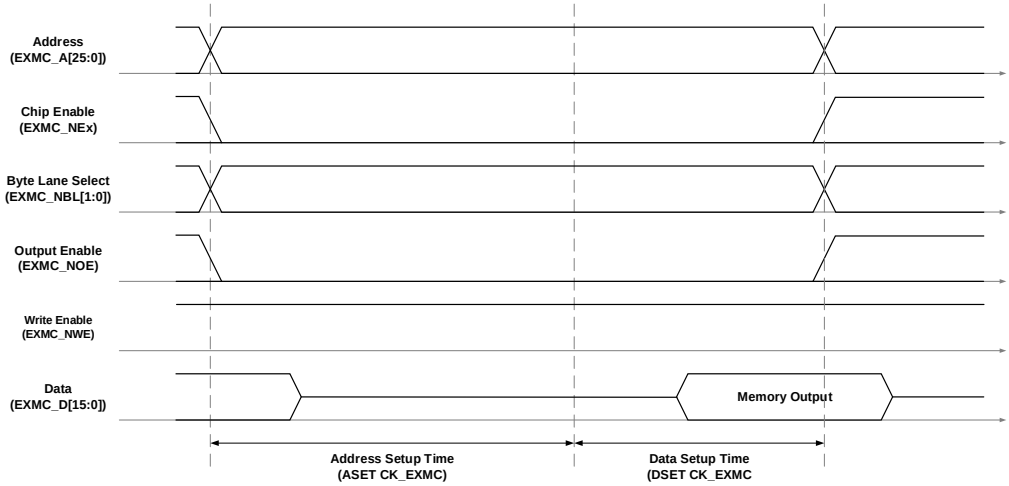


**Figure 38-8. Mode 1 write access**

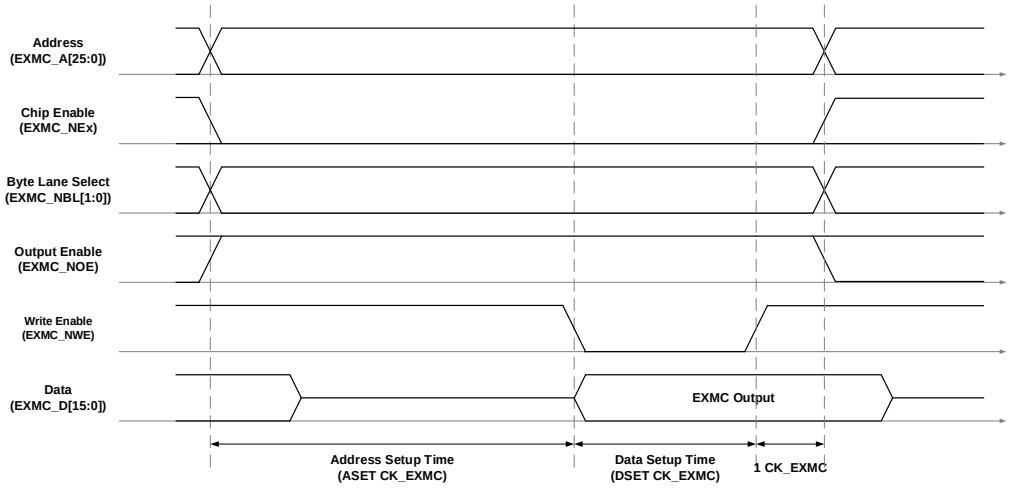



**Table 38-8. Mode 1 related registers configuration**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory and user |
| 19 | SYNCWR | 0x0 |
| 18-16 | CPS | 0x0 |
| 15 | ASYNCWTEN | Depends on memory |
| 14 | EXMODEN | 0x0 |
| 13 | NRWTEN | 0x0 |
| 12 | WEN | Depends on user |
| 11 | NRWTCFG | No effect |
| 10 | Reserved | 0x0 |
| 9 | NRWTPOL | Meaningful only when the bit 15 is set to 1 |
| 8 | SBRSTEN | 0x0 |
| 7 | Reserved | 0x1 |
| 6 | NREN | No effect |
| 5-4 | NRW | Depends on memory |
| 3-2 | NRTP | Depends on memory, except 2(Nor Flash) |
| 1 | NRMUX | 0x0 |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx** | | |
| 31-30 | Reserved | 0x0000 |
| 29-28 | ASYNCMOD | No effect |
| 27-24 | DLAT | No effect |
| 23-20 | CKDIV | No effect |
| 19-16 | BUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | Depends on memory and user (DSET+1 CK_EXMC for write, DSET CK_EXMC for read) |

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| EXMC_SNCTLx | | |
| 7-4 | AHLD | No effect |
| 3-0 | ASET | Depends on memory and user |

Mode A - SRAM/PSRAM(CRAM) OE toggling

**Figure 38-9. Mode A read access**

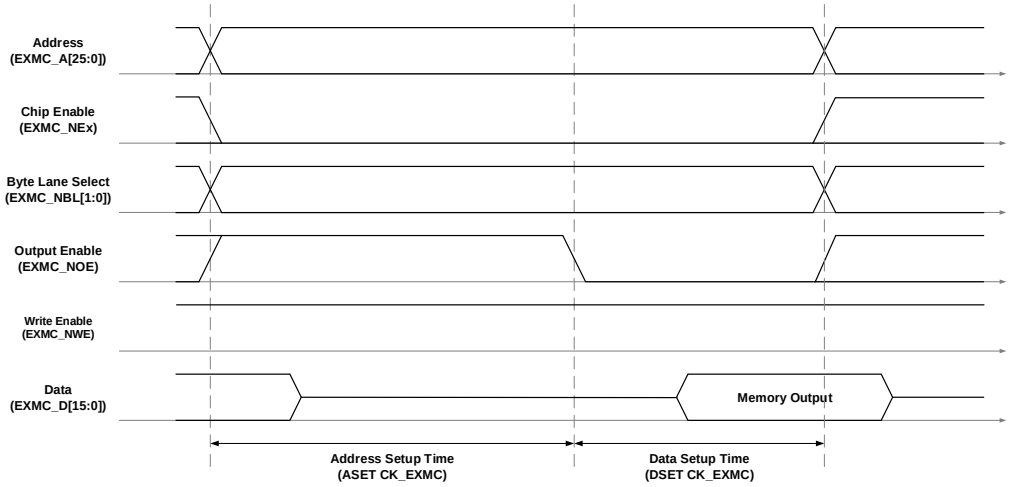
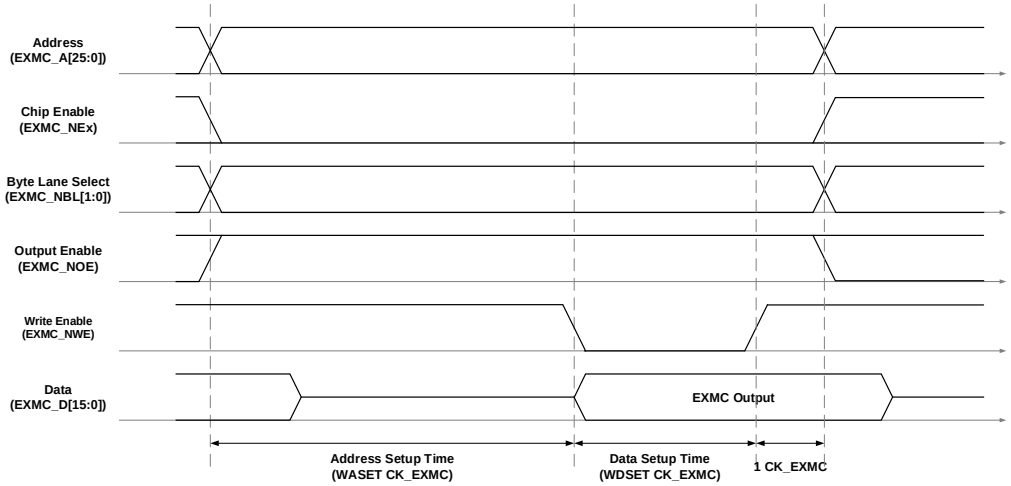



**Figure 38-10. Mode A write access**




The difference between mode A and mode 1 write timing is that read / write timing is specified by the same set of timing configuration, while mode A write timing configuration is independent of its read configuration.

**Table 38-9. Mode A related registers configuration**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| EXMC_SNCTLx | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory and user |
| 19 | SYNCWR | 0x0 |
| 18-16 | CPS | 0x0 |

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 15 | ASYNCWTEN | Depends on memory |
| 14 | EXMODEN | 0x1 |
| 13 | NRWTEN | 0x0 |
| 12 | WEN | Depends on user |
| 11 | NRWTCFG | No effect |
| 10 | Reserved | 0x0 |
| 9 | NRWTPOL | Meaningful only when the bit 15 is set to 1 |
| 8 | SBRSTEN | 0x0 |
| 7 | Reserved | 0x1 |
| 6 | NREN | No effect |
| 5-4 | NRW | Depends on memory |
| 3-2 | NRTP | Depends on memory, except 2(Nor Flash) |
| 1 | NRMUX | 0x0 |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx(Read)** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | ASYNCMOD | 0x0 |
| 27-24 | DLAT | No effect |
| 23-20 | CKDIV | No effect |
| 19-16 | BUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | Depends on memory and user (DSET+1 CK_EXMC for write, DSET CK_EXMC for read) |
| 7-4 | AHLD | No effect |
| 3-0 | ASET | Depends on memory and user |
| **EXMC_SNWTCFGx(Write)** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | WASYNCMOD | 0x0 |
| 27-20 | Reserved | 0x00 |
| 19-16 | WBUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | WDSET | Depends on memory and user |
| 7-4 | WAHLD | 0x0 |
| 3-0 | WASET | Depends on memory and user |

Mode 2/B - NOR Flash

**Figure 38-11. Mode 2/B read access**

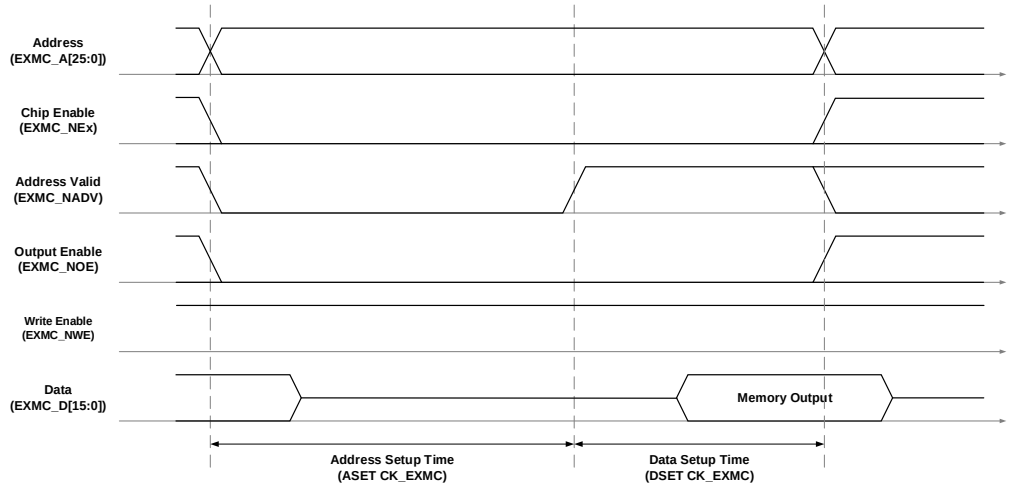



**Figure 38-12. Mode 2 write access**

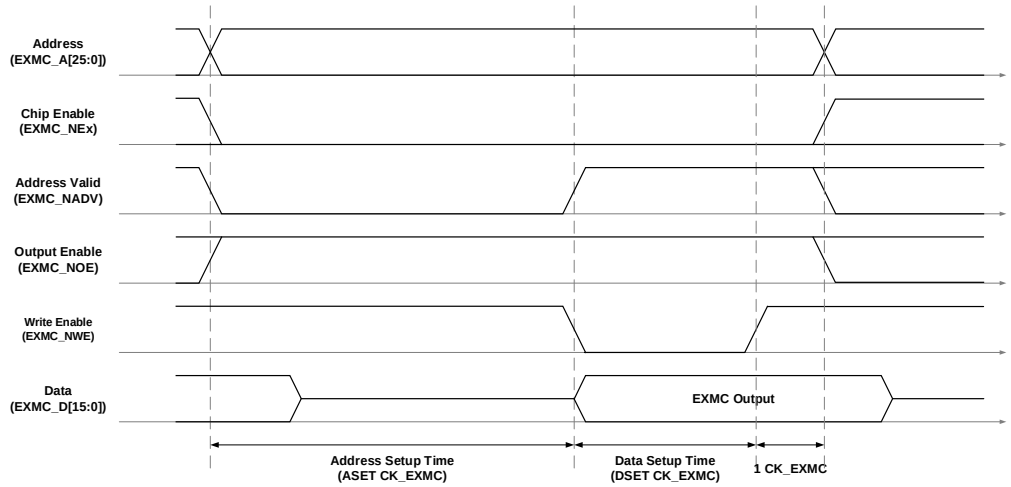



**Figure 38-13. Mode B write access**

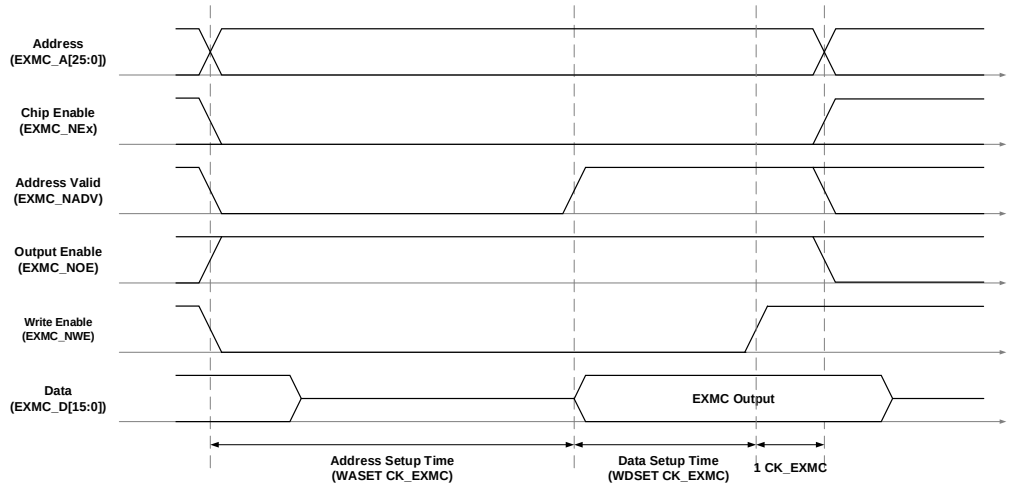



**Table 38-10. Mode 2/B related registers configuration**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| EXMC_SNCTLx(Mode 2, Mode B) | | |
| 31-21 | Reserved | 0x000 |

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx(Mode 2, Mode B)** | | |
| 20 | CCK | Depends on memory and user |
| 19 | SYNCWR | 0x0 |
| 18-16 | CPS | 0x0 |
| 15 | ASYNCWTEN | Depends on memory |
| 14 | EXMODEN | Mode 2:0x0, Mode B:0x1 |
| 13 | NRWTEN | 0x0 |
| 12 | WEN | Depends on user |
| 11 | NRWTCFG | No effect |
| 10 | Reserved | 0x0 |
| 9 | NRWTPOL | Meaningful only when the bit 15 is set to 1 |
| 8 | SBRSTEN | 0x0 |
| 7 | Reserved | 0x1 |
| 6 | NREN | 0x1 |
| 5-4 | NRW | Depends on memory |
| 3-2 | NRTP | 0x2, NOR Flash |
| 1 | NRMUX | 0x0 |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx(Read and write in mode 2,read in mode B)** | | |
| 31-30 | Reserved | 0x0000 |
| 29-28 | ASYNCMOD | Mode B:0x1 |
| 27-24 | DLAT | No effect |
| 23-20 | CKDIV | No effect |
| 19-16 | BUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | Depends on memory and user |
| 7-4 | AHLD | 0x0 |
| 3-0 | ASET | Depends on memory and user |
| **EXMC_SNWTCFGx(Write in mode B)** | | |
| 31-30 | Reserved | 0x0000 |
| 29-28 | WASYNCMOD | Mode B:0x1 |
| 27-20 | Reserved | 0x000 |
| 19-16 | WBUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | WDSET | Depends on memory and user |
| 7-4 | WAHLD | 0x0 |
| 3-0 | WASET | Depends on memory and user |

Mode C - NOR Flash OE toggling

**Figure 38-14. Mode C read access**

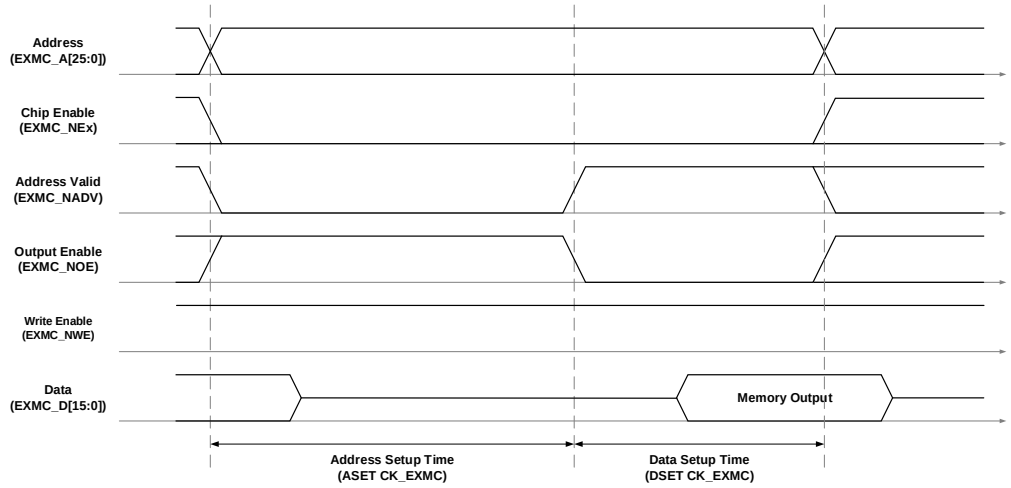
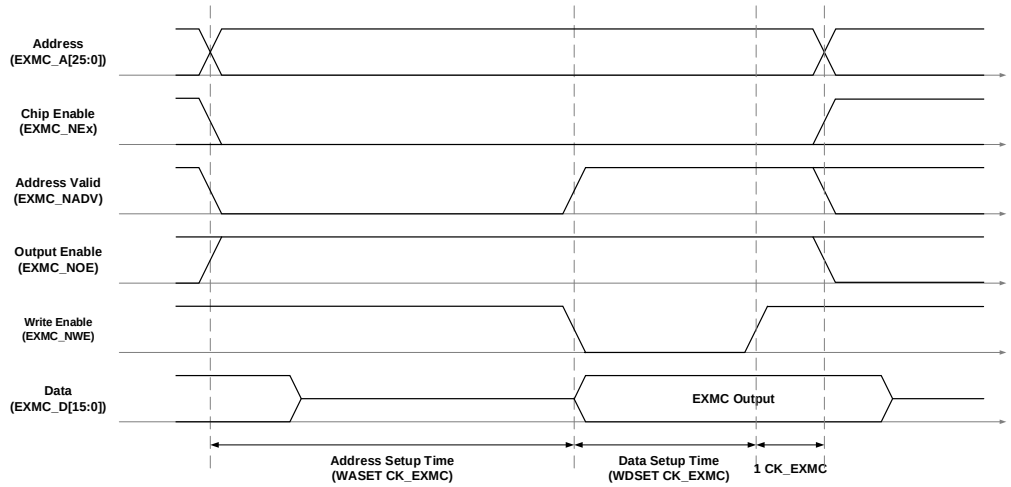



**Figure 38-15. Mode C write access**




The different between mode C and mode 1 write timing is that read / write timing is specified by the same set of timing configuration, while mode C write timing configuration is independent of its read configuration.

**Table 38-11. Mode C related registers configuration**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory and user |
| 19 | SYNCWR | 0x0 |
| 18-16 | CPS | 0x0 |
| 15 | ASYNCWTEN | Depends on memory |
| 14 | EXMODEN | 0x1 |
| 13 | NRWTEN | 0x0 |
| 12 | WEN | Depends on user |
| 11 | NRWTCFG | No effect |
| 10 | Reserved | 0x0 |

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 9 | NRWTPOL | Meaningful only when the bit 15 is set to 1 |
| 8 | SBRSTEN | 0x0 |
| 7 | Reserved | 0x1 |
| 6 | NREN | 0x1 |
| 5-4 | NRW | Depends on memory |
| 3-2 | NRTP | 0x2, NOR Flash |
| 1 | NRMUX | 0x0 |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx** | | |
| 31-30 | Reserved | 0x0000 |
| 29-28 | ASYNCMOD | Mode C:0x2 |
| 27-24 | DLAT | 0x0 |
| 23-20 | CKDIV | 0x0 |
| 19-16 | BUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | Depends on memory and user |
| 7-4 | AHLD | 0x0 |
| 3-0 | ASET | Depends on memory and user |
| **EXMC_SNWTCFGx** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | WASYNCMOD | Mode C:0x2 |
| 27-20 | Reserved | 0x00 |
| 19-16 | WBUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | WDSET | Depends on memory and user |
| 7-4 | WAHLD | 0x0 |
| 3-0 | WASET | Depends on memory and user |

Mode D - Asynchronous access with extended address

**Figure 38-16. Mode D read access**

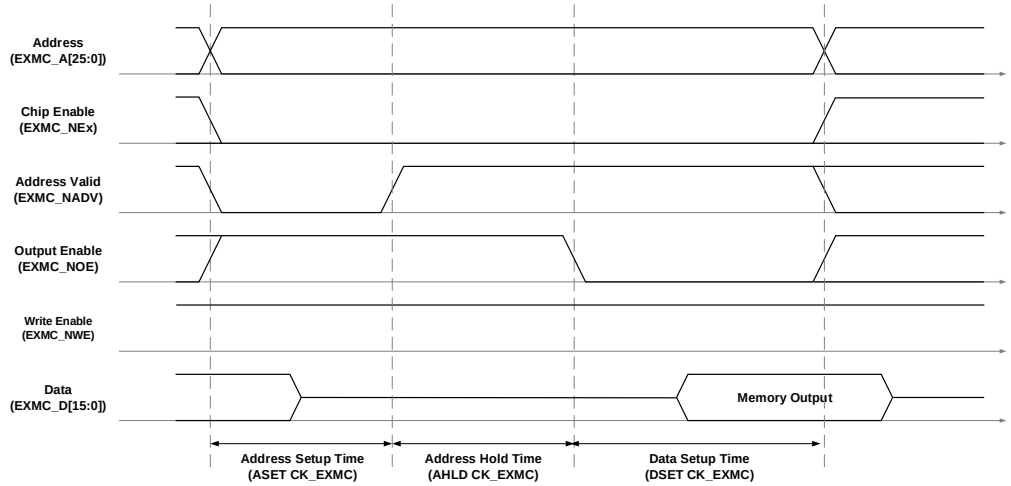


**Figure 38-17. Mode D write access**

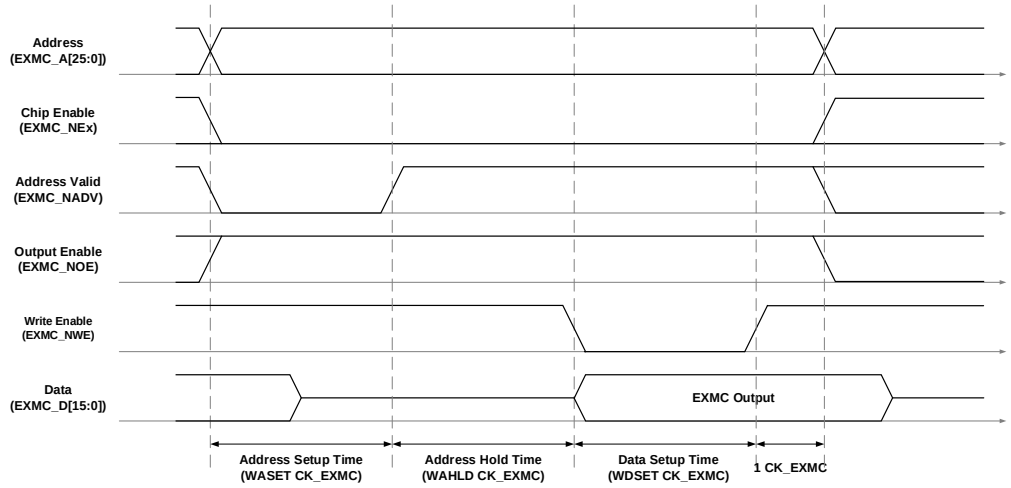



**Table 38-12. Mode D related registers configuration**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory and user |
| 19 | SYNCWR | 0x0 |
| 18-16 | CPS | 0x0 |
| 15 | ASYNCWTEN | Depends on memory |
| 14 | EXMODEN | 0x1 |
| 13 | NRWTEN | 0x0 |
| 12 | WEN | Depends on user |
| 11 | NRWTCFG | No effect |
| 10 | Reserved | 0x0 |
| 9 | NRWTPOL | Meaningful only when the bit 15 is set to 1 |
| 8 | SBRSTEN | 0x0 |
| 7 | Reserved | 0x1 |
| 6 | NREN | Depends on memory |
| 5-4 | NRW | Depends on memory |
| 3-2 | NRTP | Depends on memory |
| 1 | NRMUX | 0x0 |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | ASYNCMOD | Mode D:0x3 |
| 27-24 | DLAT | Don’t care |
| 23-20 | CKDIV | No effect |
| 19-16 | BUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | Depends on memory and user |
| 7-4 | AHLD | Depends on memory and user |

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 3-0 | ASET | Depends on memory and user |
| **EXMC_SNWTCFGx** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | WASYNCMOD | Mode D:0x3 |
| 27-20 | Reserved | 0x00 |
| 19-16 | WBUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | WDSET | Depends on memory and user |
| 7-4 | WAHLD | Depends on memory and user |
| 3-0 | WASET | Depends on memory and user |

Mode M - NOR Flash address / data bus multiplexing

**Figure 38-18. Multiplex mode read access**

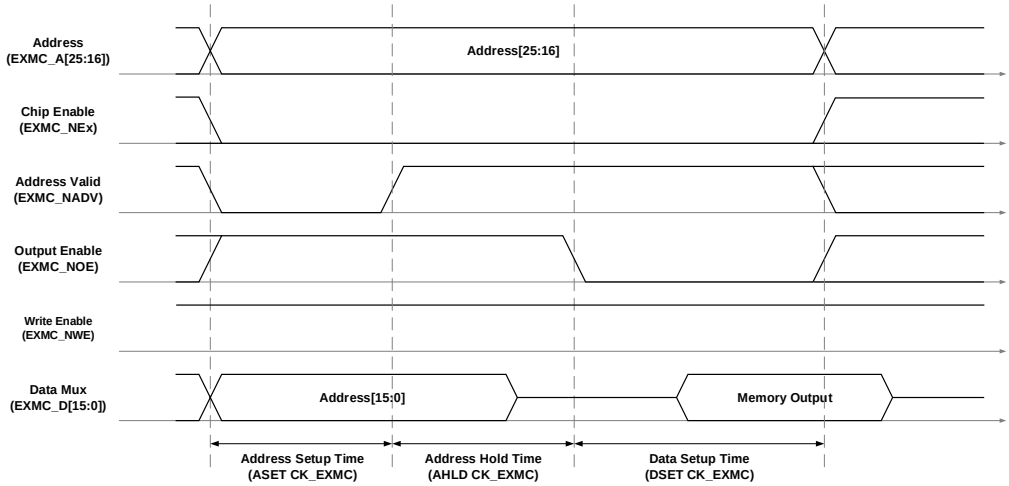



**Figure 38-19. Multiplex mode write access**

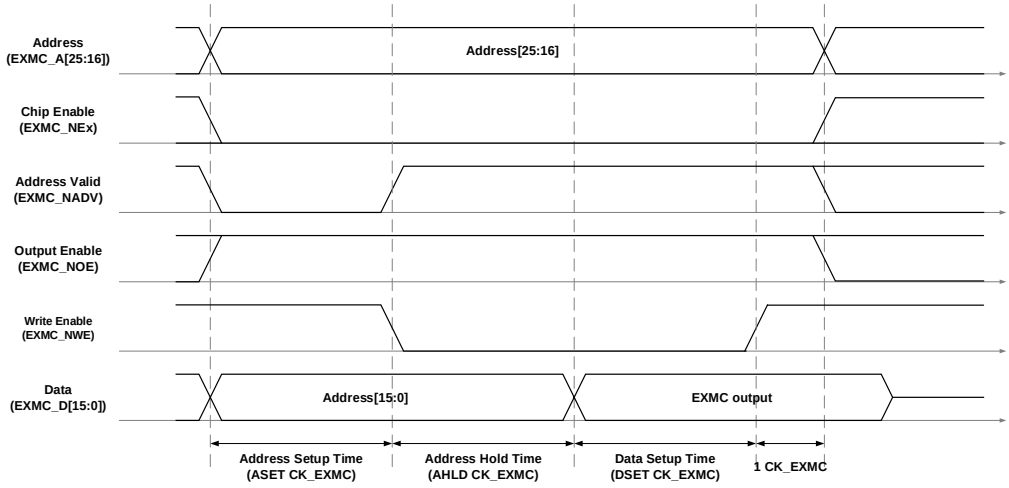


**Table 38-13. Multiplex mode related registers configuration**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory |
| 19 | SYNCWR | 0x0 |
| 18-16 | CPS | 0x0 |
| 15 | ASYNCWTEN | Depends on memory |
| 14 | EXMODEN | 0x0 |
| 13 | NRWTEN | 0x0 |
| 12 | WEN | Depends on memory |
| 11 | NRWTCFG | No effect |
| 10 | Reserved | 0x0 |
| 9 | NRWTPOL | Meaningful only when the bit 15 is set to 1 |
| 8 | SBRSTEN | 0x0 |
| 7 | Reserved | 0x1 |
| 6 | NREN | 0x1 |
| 5-4 | NRW | Depends on memory |
| 3-2 | NRTP | 0x2:NOR Flash |
| 1 | NRMUX | 0x1 |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | ASYNCMOD | 0x0 |
| 27-24 | DLAT | No effect |
| 23-20 | CKDIV | No effect |
| 19-16 | BUSLAT | Minimum time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | Depends on memory and user |
| 7-4 | AHLD | Depends on memory and user |
| 3-0 | ASET | Depends on memory and user |

**Wait timing of asynchronous communication**

Wait feature is controlled by the bit ASYNCWTEN in register EXMC_SNCTLx. During extern memory access, data setup phase will be automatically extended by the active EXMC_NWAIT signal if ASYNCWTEN bit is set. The extend time is calculated as follows:

If memory wait signal is aligned to EXMC_NOE / EXMC_NWE:

$$T_{DATA_SETUP} \geq \max T_{WAIT_ASSERTION} + 4CK_EXMC \quad (38\text{-}1)$$

If memory wait signal is aligned to EXMC_NE:

If

$$\max T_{WAIT_ASSERTION} \geq T_{ADDRES_PHASE} + T_{HOLD_PHASE} \quad (38\text{-}2)$$

$$T_{DATA_SETUP} \geq (\max T_{WAIT_ASSERTION} - T_{ADDRES_PHASE} - T_{HOLD_PHASE}) + 4CK_EXMC \quad (38\text{-}3)$$

Otherwise

$$T_{DATA_SETUP} \geq 4CK_EXMC \quad (38\text{-}4)$$

**Figure 38-20. Read access timing diagram under async-wait signal assertion**

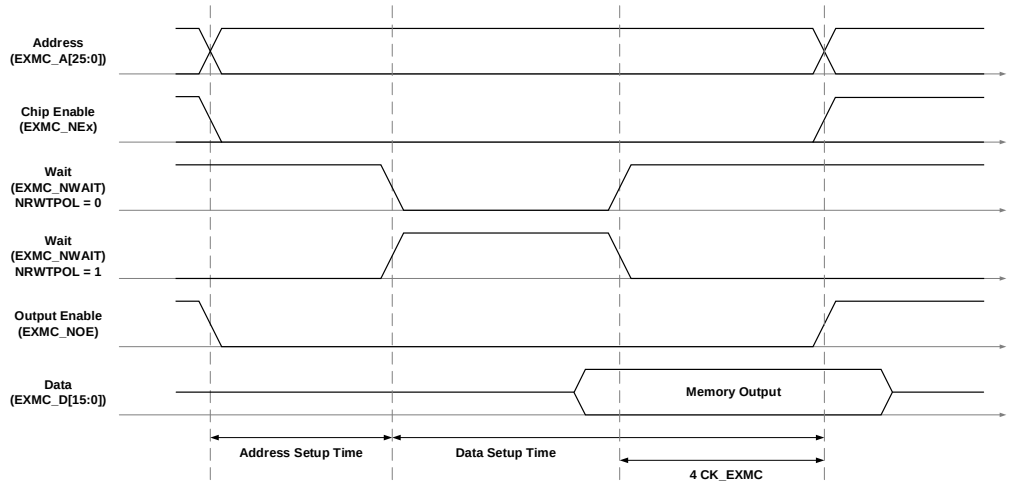
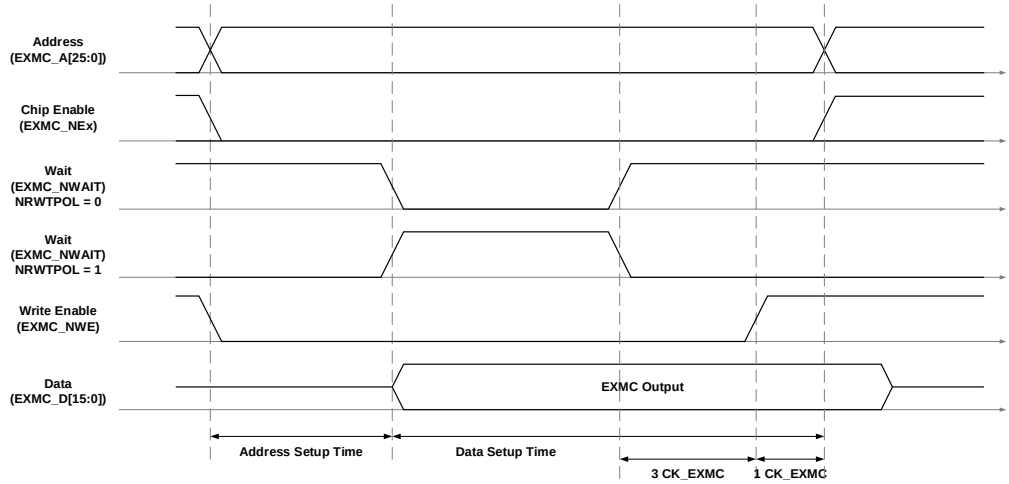



**Figure 38-21. Write access timing diagram under async-wait signal assertion**




### Synchronous access timing diagram

The relation between memory clock (EXMC_CLK) and system clock (CK_EXMC) is as follows:

$$EXMC_CLK = \frac{CK_EXMC}{CKDIV+1} \quad (38\text{-}5)$$

CKDIV is the synchronous clock divider ratio, it is configured through the CKDIV control field in the EXMC_SNTCFGx register.

1. Data latency and NOR Flash latency

Data latency (DLAT) is the number of EXMC_CLK cycles to wait before sampling the data. The relationship between data latency and NOR Flash specification's latency parameter is as

follows:

For NOR Flash's specification excluding the EXMC_NADV cycle, their relationship should be:

$$\text{NOR Flash latency} = \text{DLAT} + 2 \tag{38-6}$$

For NOR Flash's specification including the EXMC_NADV cycle, their relationship should be

$$\text{NOR Flash latency} = \text{DLAT} + 3 \tag{38-7}$$

2. Data wait

Users should guarantee that EXMC_NWAIT signal's behavior matches that of the external device. This signal is configured through the EXMC_SNCTLx registers, it is enabled by the NRWTEN bit, and the active timing could be one data cycle before the wait state or active during the active state by the configuration NRWTCFG bit, while the wait signal's polarity is set by the NRWTPOL bit.

In NOR Flash synchronous burst access mode, when NRWTEN bit in EXMC_SNCTLx register is set, EXMC_NWAIT signal will be detected after a period of data latency. If EXMC_NWAIT signal detected as valid, wait cycles will be inserted until EXMC_NWAIT becomes invalid.

- The valid polarity of EXMC_NWAIT:

NRWTPOL= 1: valid level of EXMC_NWAIT signal is high.

NRWTPOL= 0: valid level of EXMC_NWAIT signal is low.

- In synchronous burst mode, EXMC_NWAIT signal has two kinds of configurations:

NRWTCFG = 1: When EXMC_NWAIT signal is active, current cycle data is not valid.

NRWTCFG = 0: When EXMC_NWAIT signal is active, the next cycle data is not valid. It is the default state after reset.

During wait-state inserted via the EXMC_NWAIT signal, the controller continues to send clock pulses to the memory, keep the chip select and output signals availably, and ignore the invalid data signal.

3. Automatic burst split at CRAM page boundary

Crossing page boundary burst access is prohibited in CRAM 1.5, an automatic burst split functionality is implemented by the EXMC. To guarantee correct burst split operation, users should specify CRAM page size by configuring the CPS bit in EXMC_SNCTLx register to inform the EXMC when this functionality should be performed.

4. Mode SM - Single burst transmission

For synchronous burst transmission, if the needed data of AXI is 16-bit, EXMC will perform a burst transmission whose length is 1. If the needed data of AXI is 32-bit, EXMC will make the transmission divided into two 16-bit transmissions, that is, EXMC performs a burst transmission whose length is 2.

For other configurations please refers to ***Table 38-5. EXMC bank 0 supports all transactions***.

Synchronous mux burst read timing - NOR, PSRAM (CRAM)

**Figure 38-22. Read timing of synchronous multiplexed burst mode**

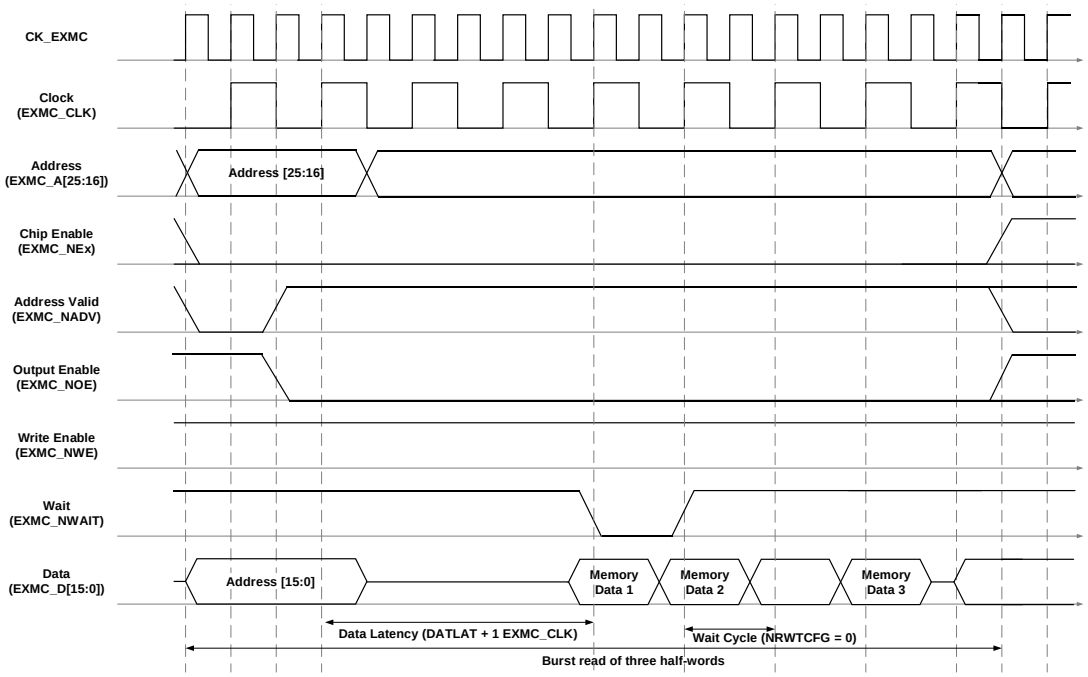



**Table 38-14. Timing configurations of synchronous multiplexed read mode**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory |
| 19 | SYNCWR | No effect |
| 18-16 | CPS | Depends on memory |
| 15 | ASYNCWTEN | 0x0 |
| 14 | EXMODEN | 0x0 |
| 13 | NRWTEN | Depends on memory |
| 12 | WEN | No effect |
| 11 | NRWTCFG | Depends on memory |
| 10 | Reserved | 0x0 |
| 9 | NRWTPOL | Depends on memory |
| 8 | SBRSTEN | 0x1, burst read enable |
| 7 | Reserved | 0x1 |
| 6 | NREN | Depends on memory |
| 5-4 | NRW | 0x1 |
| 3-2 | NRTP | Depends on memory, 0x1/0x2 |
| 1 | NRMUX | 0x1, Depends on memory and users |
| 0 | NRBKEN | 0x1 |

| EXMC_SNTCFGx(Read) | | |
|---|---|---|
| 31-30 | Reserved | 0x0 |
| 29-28 | ASYNCMOD | 0x0 |
| 27-24 | DLAT | Data latency |
| 23-20 | CKDIV | The figure above: 0x1,EXMC_CLK=2 CK_EXMC |
| 19-16 | BUSLAT | Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge |
| 15-8 | DSET | No effect |
| 7-4 | AHLD | No effect |
| 3-0 | ASET | No effect |

Mode SM –Synchronous mux burst write timing – NOR, PSRAM (CRAM)

**Figure 38-23. Write timing of synchronous multiplexed burst mode**

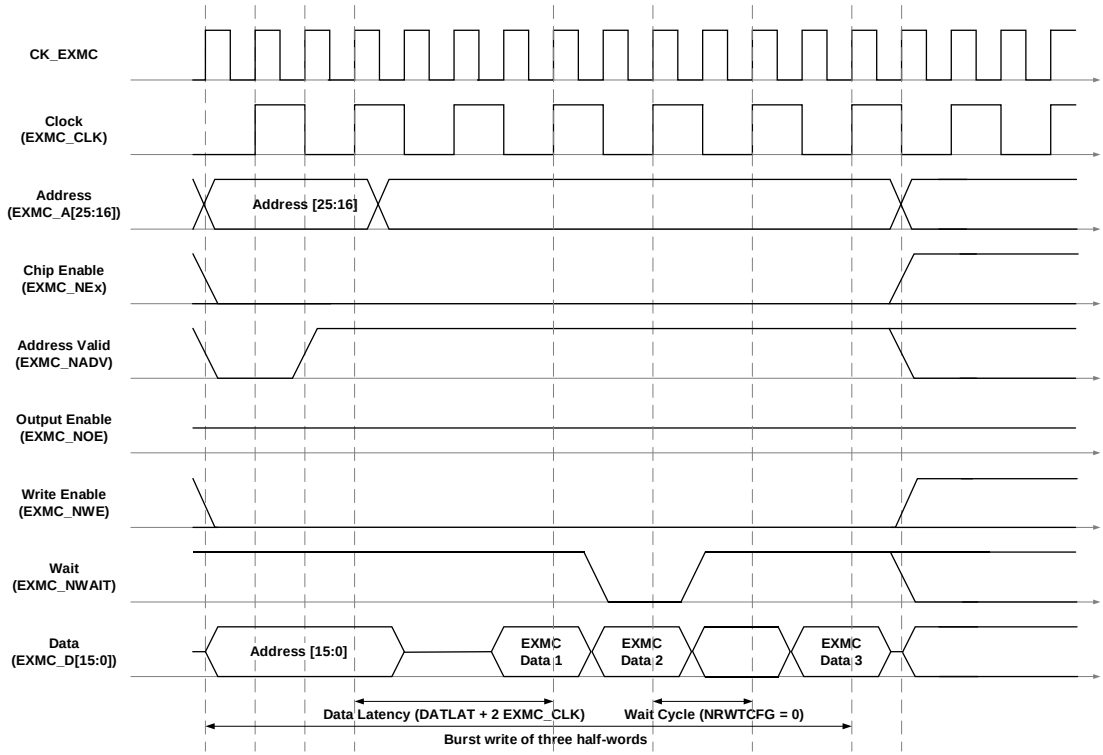



**Table 38-15. Timing configurations of synchronous multiplexed write mode**

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| EXMC_SNCTLx | | |
| 31-21 | Reserved | 0x000 |
| 20 | CCK | Depends on memory |
| 19 | SYNCWR | 0x1, synchronous write enable |
| 18-16 | CPS | Depends on memory |
| 15 | AYSNCWAIT | 0x0 |
| 14 | EXMODEN | 0x0 |
| 13 | NRWTEN | Depends on memory |
| 12 | WEN | 0x1 |
| 11 | NRWTCFG | 0x0(Here must be zero) |

| Bit Position | Bit Name | Reference Setting Value |
|---|---|---|
| **EXMC_SNCTLx** | | |
| 10 | Reserved | 0x0 |
| 9 | NTWTPOL | Depends on memory |
| 8 | SBRSTEN | No effect |
| 7 | Reserved | 0x1 |
| 6 | NREN | Depends on memory |
| 5-4 | NRW | 0x1 |
| 3-2 | NRTP | 0x1 |
| 1 | NRMUX | 0x1, Depends on users |
| 0 | NRBKEN | 0x1 |
| **EXMC_SNTCFGx(Write)** | | |
| 31-30 | Reserved | 0x0 |
| 29-28 | ASYNCMOD | 0x0 |
| 27-24 | DLAT | Data latency |
| 23-20 | CKDIV | The figure above: 0x1, EXMC_CLK=2 CK_EXMC |
| 19-16 | BUSLAT | No effect |
| 15-8 | DSET | No effect |
| 7-4 | AHLD | No effect |
| 3-0 | ASET | No effect |

### 38.3.7. NAND flash controller

EXMC has partitioned Bank2 as NAND Flash access field, Bank1 and Bank3 are reserved. Each bank has its own set of control register for access timing configuration. 8- and 16-bit NAND Flash are supported. An ECC hardware is provided for the NAND Flash controller to ensure the robustness of data transfer and storage.

#### NAND flash interface function

**Table 38-16. 8-bit or 16-bit NAND interface signal**

| EXMC Pin | Direction | Functional description |
|---|---|---|
| EXMC_A[17] | Output | NAND Flash address latch (ALE) |
| EXMC_A[16] | Output | NAND Flash command latch (CLE) |
| EXMC_D[7:0]/ EXMC_D[15:0] | Input /Output | 8-bit multiplexed, bidirectional address/data bus |
| | | 16-bit multiplexed, bidirectional address/data bus |
| EXMC_NCE | Output | Chip select |
| EXMC_NOE(NRE) | Output | Output enable |
| EXMC_NWE | Output | Write enable |
| EXMC_NWAIT/ EXMC_INT | Input | NAND Flash ready/busy input signal to the EXMC |

### Supported memory access mode

**Table 38-17. Bank2 of EXMC support the memory and access mode**

| Memory | Mode | R/W | AXI transaction size | Comments |
|---|---|---|---|---|
| 8-bit NAND | Async | R | 8 | |
| | Async | W | 8 | |
| | Async | R | 16 | Automatically split into 2 EXMC accesses |
| | Async | W | 16 | |
| | Async | R | 32 | Automatically split into 4 EXMC accesses |
| | Async | W | 32 | |
| | Async | R | 64 | Automatically split into 8 EXMC accesses |
| | Async | W | 64 | |
| 16-bit NAND | Async | R | 8 | |
| | Async | W | 8 | Not support this operation |
| | Async | R | 16 | |
| | Async | W | 16 | |
| | Async | R | 32 | Automatically split into 2 EXMC accesses |
| | Async | W | 32 | |
| | Async | R | 64 | Automatically split into 4 EXMC accesses |
| | Async | W | 64 | |

### NAND flash controller timing

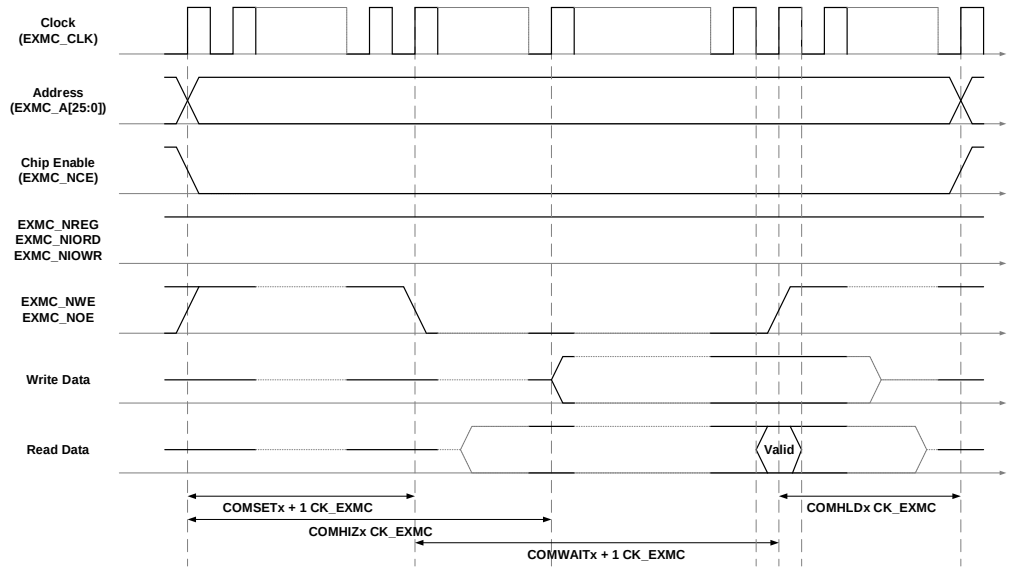
EXMC can generate the appropriate signal timing for NAND Flash and other devices. Each bank has a corresponding register to manage and control the external memory, such as EXMC_NCTL, EXMC_NINTEN, EXMC_NCTCFG, EXMC_NATCFG and EXMC_NECC. Among these registers, EXMC_NCTCFG, EXMC_NATCFG registers contain four timing parameters individually which are configured according to user specification and features of the external memory.

**Table 38-18. NAND flash programmable parameters**

| Programmable parameter | W/R | Unit | Functional description | NAND Flash | |
|---|---|---|---|---|---|
| | | | | Min | Max |
| High impedance time of the memory data bus (HIZ) | W/R | CK_EXMC | Time to keep the data bus high impedance after starting write operation | 1 | 255 |
| Memory hold time (HLD) | W/R | CK_EXMC | The number of CK_EXMC clock cycles to keep address valid after sending the command. In write mode, it is also data hold time. | 1 | 254 |
| Memory wait time (WAIT) | W/R | CK_EXMC | Minimum duration of sending command | 2 | 255 |
| Memory setup time (SET) | W/R | CK_EXMC | The number of CK_EXMC clock cycles to build address before sending command | 1 | 256 |

The ***Figure 38-24. Access timing of common memory space of NAND flash controller*** shows the programmable parameters which are defined in the common memory space operations. The programmable parameters of Attribute memory space are defined as well.

**Figure 38-24. Access timing of common memory space of NAND flash controller**




## NAND flash operation

When EXMC sends command or address to NAND Flash, it needs to use the command latch signal (A[16]) or address latch signal (EXMC_A[17]), namely, the CPU needs to perform write operation in particular address.

Example: NAND Flash read operation steps:

1. Configure EXMC_NCTL and EXMC_NCTCFG register. When pre-waiting is needed, EXMC_NATCFG has to be configured.

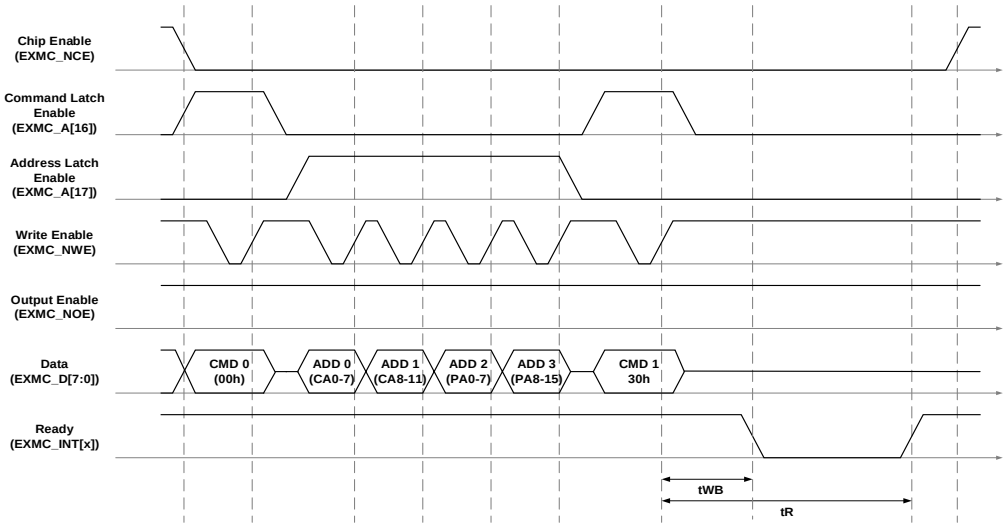
2. Send the command of NAND Flash read operation to the common space. Namely, during the valid period of EXMC_NCE and EXMC_NWE, when EXMC_CLE (EXMC_A[16]) becomes valid (high level), data on the I/O pins is regarded as a command by NAND Flash.
3. Send the start address of read operation to the common space. During the valid period of EXMC_NCE and EXMC_NWE, when EXMC_ALE (EXMC_A[17]) becomes valid (high level), the data on the I/O pins is regarded as an address by NAND Flash.
4. Waiting for NAND ready signal. In this period, NAND controller will maintain EXMC_NCE valid.
5. Read data byte by byte from the data area of the common space.
6. If new commands or address haven't been written, data of the next page can be read out automatically. You can also read the data of the next page by going to step 3 and then writing a new address or writing a new command and address in step 2.

### NAND flash pre-wait functionality

Some NAND Flash requires that the controller should wait for NAND Flash to be ready, after the first command byte and following the address bytes are send, and some EXMC_NCE-sensitive NAND Flash also requires that the EXMC_NCE must remain valid before it is ready.

Taking TOSHIBA128 M x 8 bit NAND Flash as an example:

**Figure 38-25. Access to none "NCE don't care" NAND Flash**




1. Write CMD0 into NAND Flash bank common space command area.
2. Write ADD0 into NAND Flash bank common space address area.
3. Write ADD1 into NAND Flash bank common space address area.
4. Write ADD2 into NAND Flash bank common space address area.
5. Write ADD3 into NAND Flash bank common space address area.
6. Write CMD1 into NAND Flash bank attribute space command area.

In step 6, EXMC uses the operation timing defined in EXMC_NATCFG register. After a period of ATTHLD, NAND Flash waits for EXMC_INT signal to be busy, and the time period of ATTHLD should be greater than $t_{WB}$ ($t_{WB}$ is defined as the time from EXMC_NWE high to

EXMC_INT low). For NCE-sensitive NAND Flash, after the first command byte following address bytes has been entered, EXMC_NCE must remain low until EXMC_INT goes from low to high. The ATTHLD value of attribute space can be set in EXMC_NATCFG register to meet the timing requirements of $t_{WB}$. MCU can use the attribute space timing when writing the first command byte following address bytes to the NAND Flash device. In other times, the MCU must use the common space timing.

#### NAND flash ECC calculation module

An ECC calculation hardware is implemented in Bank2 respectively. Users can choose page size according to the ECCSZ control field in the EXMC_NCTL register. ECC offers one bit error correction and two bits errors detection.

When NAND memory block is enabled, ECC module will detect EXMC_D[15:0], EXMC_NCE and EXMC_NWE signals. When a data size of ECCSZ has been read or written, software must read the calculated ECC in the EXMC_NECC register. When a recalculation of ECC is needed, software must clear the EXMC_NECC register value by resetting ECCEN bit of EXMC_NCTL register to zero, and then restart ECC calculation by setting the ECCEN bit of EXMC_NCTL to 1.

### 38.3.8. SDRAM controller

#### Characteristics

- Two independent SDRAM devices.
- 8-,16- or 32-bit data bus width.
- Up to 13-bits Row Address, 11-bits Column Address and 2-bits internal banks address.
- Supported memory size: 4x16Mx32bit(256 MB), 4x16Mx16bit (128 MB) and 4x16Mx8bit (64 MB).
- AXI double word, word, half-word and byte access.
- Independent Chip Select control for each memory device.
- Independent configuration for each memory device.
- Write enable and byte lane select outputs.
- Automatic row and bank boundary management.
- Multi-device Ping-Pong access.
- SDRAM clock configured as $f_{CK_EXMC}/2$, $f_{CK_EXMC}/3$, $f_{CK_EXMC}/4$ or $f_{CK_EXMC}/5$.
- Programmable timing parameters.
- Automatic Refresh operation with programmable Refresh rate.
- SDRAM power-up initialization by software.
- CAS latency of 1,2,3.
- Write Data FIFO with 16 x35-bit depth.
- Write Address FIFO with 16x31-bit depth.
- Cacheable Read Data FIFO with 6 x32-bit depth.
- Cacheable Read address FIFO with 6 x14-bit depth.
- Adjustable read data sample clock.

- Self-refresh mode.
- Power-down mode.

### SDRAM overview

Synchronous dynamic random-access memory (SDRAM) is a dynamic random access memory (DRAM) whose external interface is coordinated by a synchronous external clock, this clock is provided by the EXMC through the SDRAM clock (EXMC_SDCLK) pin, and its frequency could be configured to be $f_{CK_EXMC}/2$, $f_{CK_EXMC}/3$, $f_{CK_EXMC}/4$ or $f_{CK_EXMC}/5$ according to the SDRAM clock configuration bit (SDCLK) in the EXMC_SDCTLx register. Commands and data are always latched by the SDRAM on the rising edge of EXMC_SDCLK and change on its falling edge.

SDRAM is divided into several independent sections of memory called banks, allowing the device to operate on several memory access commands in an interleaved fashion to achieve greater concurrency and higher data transfer rates. Each bank could be pictured as a matrix with each entry size equals to the memory data bus width, and the size of the matrix is the number of rows by the number of columns, thus each memory bank size could be calculated as entry_size*rows*columns. When interfacing with SDRAM, users should specify the memory dimension configurations to EXMC through NBK, SDW, RAW and CAW bits in the SDRAM control register EXMC_SDCTLx.

Due to the volatile nature of SDRAM, periodic refresh cycle is necessary to maintain the stored information. Two refresh mode could be selected, self-refresh and auto-refresh mode. Self-refresh mode is typically set in low power mode when EXMC is suspended, refresh is provided by the SDRAM and timed by its internal counter. In auto-refresh mode, refresh command is provided by the EXMC, this is necessary because SDRAM must maintain the stored information during an on-going transaction, refresh commands are issued periodically on the data bus timed by ARINTV bits in EXMC_SDARI register, the number of consecutive refresh needed is configured through NARF bits in EXMC_SDCMD register. Refresh command always take precedence over other command or read / write operation to guarantee correct data storage, when memory access occurs simultaneously with refresh command, memory access is buffered and processed when refresh command is completed. If a new refresh command occurs while the previous refresh command is buffered, a refresh error flag (REIF) is raised in EXMC_SDSTAT register, and interrupt is generated if REIE is set and cleared by setting REC bit in EXMC_SDARI register.

CAS latency defines the delay in clock cycles, between the issued read command and the availability of the first piece of data form SDRAM. CAS latency is configured by the CL bits in the EXMC_SDCTLx register.

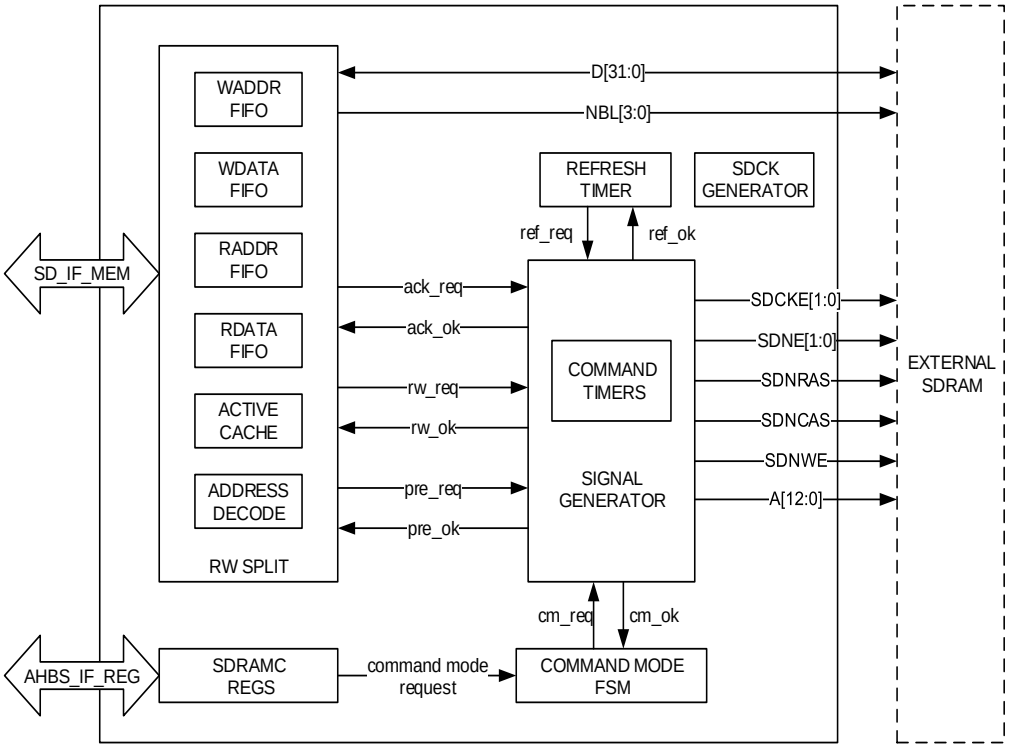
Mode register is used to define the specific operating mode of SDRAM, including burst length, burst type, CAS latency, and write mode. Users should refer to the SDRAM's specification for correct configuration. Once the operating mode has been decided, users should write the mode register content to MRC bits and issuer load mode register command through CMD bits in EXMC_SDCMD register. Load mode register command should be performed before read or write access, otherwise SDRAM might not work as expected.

### SDRAM controller overview

The synchronous dynamic random-access memory controller (SDRAMC) block acts as the interface between MCU and SDRAM memory. It translates AXI transactions into the appropriate SDRAM protocol, and meanwhile, makes sure the access timing requirements of the external SDRAM devices are satisfied by the configuration of EXMC_SDTCFG register.

SDRAMC could be divided into 4 sub-modules, the read / write split, control registers, finite state machine, and signal generator. Two pairs of FIFO is implemented to increase memory access efficiency, one pair for write address and data, the other pair for read address and data. SDRAMC's block diagram is shown in ***Figure 38-26. SDRAM controller block diagram***.

**Figure 38-26. SDRAM controller block diagram**




The signal generator handles requests from command mode FSM, Refresh timer and the RW split module.

The command timers are composed by timing counters which take case the timing specification of the SDRAM protocol.

SDRAM commands are issued by the SDRAM controller interface in the following pattern.

**Table 38-19. SDRAM command truth table**

| SDNE | NRAS | NCAS | SDNWE | A[n] | A[10] | A[m] | Command |
|---|---|---|---|---|---|---|---|
| H | X | X | X | X | X | X | Command inhibit (No operation) |
| L | H | H | H | X | X | X | No operation |

| SDNE | NRAS | NCAS | SDNWE | A[n] | A[10] | A[m] | Command |
|---|---|---|---|---|---|---|---|
| L | H | H | L | X | X | X | Burst Terminate |
| L | H | L | H | Bank | L | Col | Burst read from current row |
| L | H | L | H | Bank | H | Col | Burst read from current row, precharge when done |
| L | H | L | L | Bank | L | Col | Burst write to current row |
| L | H | L | L | Bank | H | Col | Burst write to current row, precharge when done |
| L | L | H | H | Bank | Row | Row | Active, open row for read / write |
| L | L | H | L | Bank | L | X | Precharge, close current row of the selected bank |
| L | L | H | L | X | H | X | Precharge all, close current row of all banks |
| L | L | L | H | X | X | X | Auto-refresh when SDCKE = 1<br>Self-refresh when SDCKE = 0 |
| L | L | L | L | L | Mode | Mode | Load mode register |

## SDRAM controller operation sequence

### IO configuration

SDRAMC IO port must be configured first to interface with external SDRAM, otherwise it is left as general purpose IOs, and could be utilized by other modules. IO ports related to SDRAM operations are summarized in ***Table 38-20. IO definition of SDRAM controller***.

**Table 38-20. IO definition of SDRAM controller**

| Signal | Direction | Description |
|---|---|---|
| EXMC_SDCLK | O | SDRAM memory clock |
| EXMC_SDCKE[0] | O | Clock enable for SDRAM memory 0 |
| EXMC_SDCKE[1] | O | Clock enable for SDRAM memory 1 |
| EXMC_SDNE[0] | O | Chip select for SDRAM memory 0, active low |
| EXMC_SDNE[1] | O | Chip select for SDRAM memory 1, active low |
| EXMC_NRAS | O | Row address strobe, active low |
| EXMC_NCAS | O | Column address strobe, active low |
| EXMC_SDNWE | O | Write enable, active low |
| EXMC_A[12:0] | O | Address |
| EXMC_A[15:14] | O | Bank address |
| EXMC_D[31:0] | I/O | Read / write Data |
| EXMC_NBL[3:0] | O | Write data mask, the Low byte lane is accessed |

### Controller initialization

Users should follow procedure to initialize the SDRAM controller, the initialization sequence could be applied to a single SDRAM, or two SDRAM simultaneously. This choice is made by the device selection bits DS0 and DS1 in EXMC_SDCMD register. Initialization sequence must be performed before any read / write memory access, otherwise, EXMC's behavior is

not guaranteed.

1. Control parameter specification: SDRAM control register EXMC_SDCTLx should be programed first to specify the external memory dimension, clock configuration, and read / write strategy.
2. Timing parameter specification: SDRAM timing configuration register EXMC_SDTCFGx should be programed according to external SDRAM data sheet for SDRAM controller to keep pace with the operation of the external SDRAM. RPD and ARFD must be programed in EXMC_SDTCFG0, those corresponding bit position in EXMC_SDTCFG1 are reserved.
3. Enable SDCLK: SDCLK enable command should be issued to the corresponding SDRAM devices, this is done by writing 0b001 to the CMD bits in the EXMC_SDCMD register, DS0 and DS1 selected which device will accept the command and start receiving EXMC_SDCLK.
4. Power-up delay: typical delay is around 100us.
5. Precharge all: A precharge all command should be issued to reset all the SDRAM memory banks to their idle state, waiting for subsequent operation. This is done by writing 0b010 to the CMD bits in the EXMC_SDCMD register, DS0 and DS1 defines which SDRAM device will receive this command.
6. Set auto-refresh: Auto-refresh command is sent by writing 0b011 in the CMD bits in EXMC_SDCMD register. Users should also specify the number of consecutive refresh command to issue each time by configuring the NARF bits, this configuration is requested by SDRAM specification, it is also where users should refer to.DS0 and DS1 defines which SDRAM device will receive this command.
7. Mode register configuration: Mode register is programed by writing the mode register content in MRC bits in EXMC_SDCMD register, mode register specifies the operating mode of SDRAM, such modes include burst length, burst type, CAS latency, and write mode. Users should refer to the SDRAM's specification for correct configuration. CAS latency should be the same as the CL bits in EXMC_SDCTLx register, and burst length of 1 must be selected, otherwise SDRAMC's behavior is not guaranteed. If the mode register contents are different for both SDRAM devices, this step should be repeated, targeting one device a time by the DS0 and DS1 configuration.
8. Set auto-refresh rate: Auto-refresh rate corresponds to the time between refresh cycles, users must ensure that this time period match that of the SDRAM specification.

Now SDRAMC is ready to proceed with memory access. If system reset happens, the initialization sequence must be repeated. Initialization must be performed at least once before SDRAM read / write access.

**Precharge**

When the memory controller needs to access a different row, it must first return that bank's sense amplifiers to an idle state, ready to sense the next row. This is known as a precharge operation, or deactivating the row. A precharge may be commanded explicitly by the precharge all command, or it may perform automatically at the conclusion of a read or write operation. There is a minimum time, the row precharge delay (RPD), which must elapse

before that bank is fully idle and it may receive another activate command.

**Activate**

The activate command activates an idle bank. It presents a 2-bit bank address EXMC_A[15:14] and a 13-bit row address EXMC_A[12:0], and causes a read of that row into the bank's array of 16,384 column sense amplifiers. This also known as opening the row. This operation has the side effect of refreshing the dynamic memory storage cells of that row.

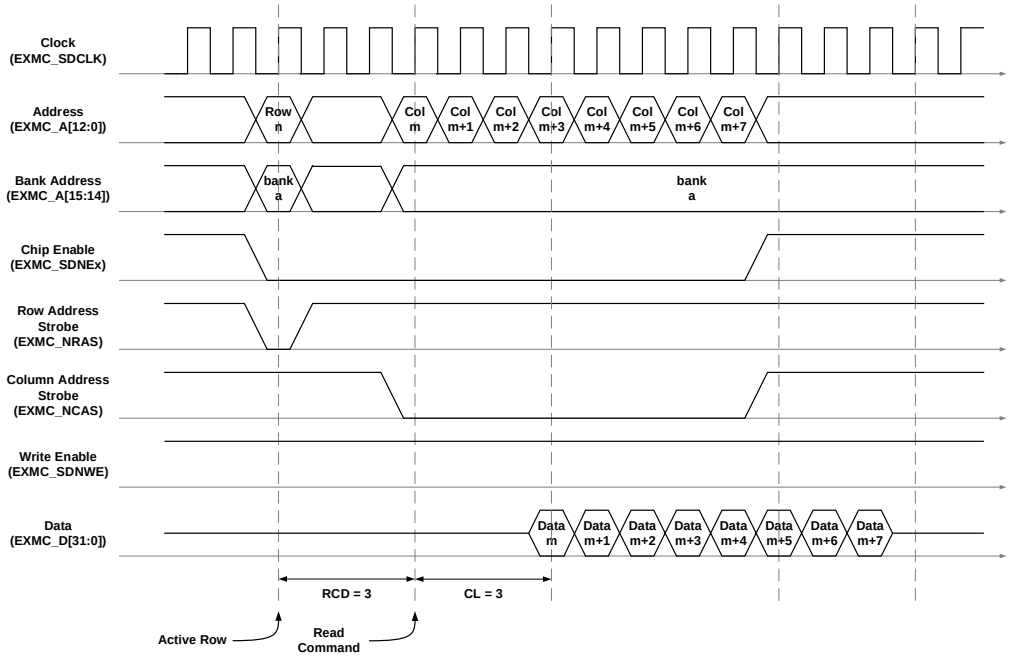
Once the row has been activated, read / write commands are possible to that row. Activation requires a minimum amount of time, called the row-to-column delay (RCD) before read / write to it may occur. This time, rounded up to the next multiple of the clock period, specifies the minimum number of wait cycles between an active command and a read / write command. During these wait cycles, additional commands may be sent to other banks, because each bank operates completely independently.

**Read / write access**

SDRAMC can translate AXI single and burst read operation into single memory access. SDRAMC always keeps track of the activated row number in order to perform consecutive read access. If the next read location is in the same row or another active row, read access is proceeded without interruption, else a precharge command is issued to deactivate the current row, followed by the activation of the row where the next read access is targeted, and then the read access is performed. A read FIFO is design to cache the read data during CAS latency and pipe line delay (PIPED), Burst read (BRSTRD) must be set in order to enable the FIFO.

The ***Figure 38-27. Burst read operation*** shows a burst read access to an in active row, a row activation command is issued before read access. If read operation were performed on an active row, row address strobe is not necessary, only column address strobe is needed.

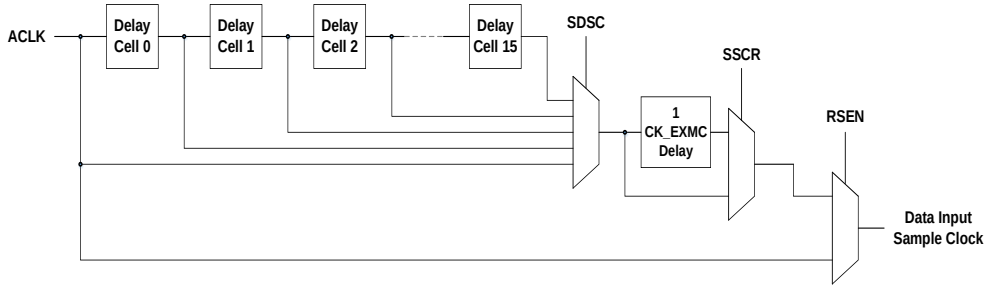
**Figure 38-27. Burst read operation**




An internal generated clock, which has an adjustable delay from the CK_EXMC can be used to sample read data from external memories. This clock can be helpful when the read data can't be sampled correctly by CK_EXMC. When this clock is enabled, the read data will be firstly stored in an asynchronous FIFO before returned to the AXI bus. Additional delays of about 2~3 CK_EXMC may be brought into the reading command process.

A clock delay chain module is added after the CK_EXMC input to the signal generator, this delayed clock is used as the sampling clock of the input data. The delay chain is controlled by the EXMC_SDRSCTL register, RSEN bit select whether the CK_EXMC output is delay at all, SSCR bit select whether 1 additional CK_EXMC cycle is added to the total delay, and SDSC select how many delay cells is add, the number of delay cell could be added is within 0 and 15. The following diagram shows how delay chain is added.

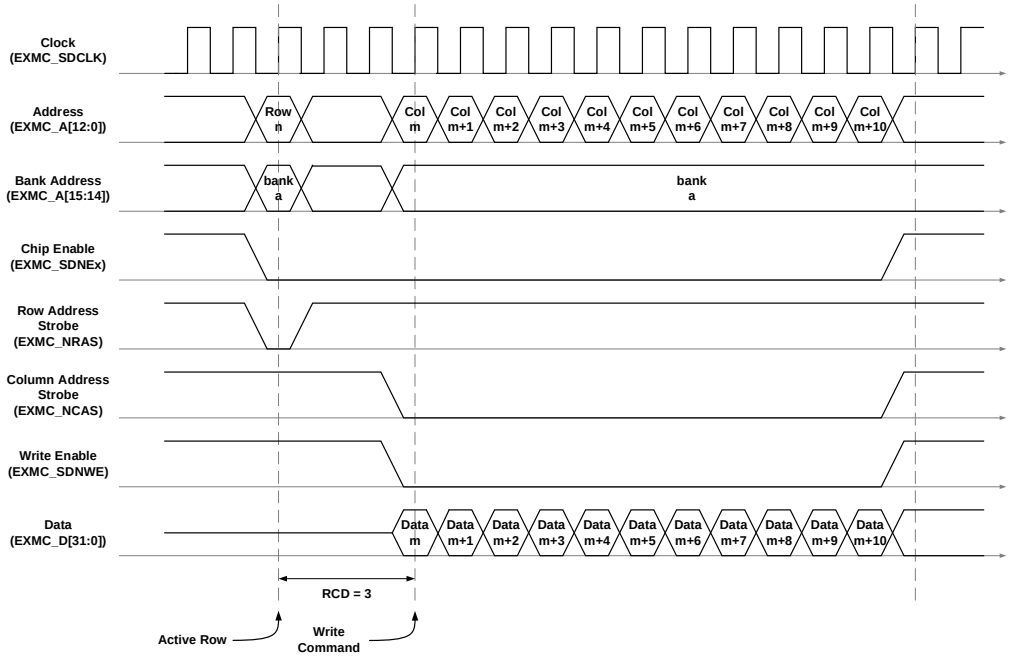
**Figure 38-28. Data sampling clock delay chain**




SDRAMC can translate AXI single and burst write operation into single memory access. Write protection must be disabled by resetting WPEN bit in EXMC_SDCTLx register. SDRAMC always keeps track of the activated row number in order to perform consecutive write access. If next write location is in the same row or another active row, write access is proceeded without interruption, else a precharge command is issued to deactivate the current row, followed by the activation of the row where the next write access is targeted, and then the write access is performed.

The ***Figure 38-29. Burst write operation*** shows a write burst access to an inactive row, a row activation command is issued before write access. If write operations were performed on an active row, row address strobe is not necessary, only column address strobe is needed.

**Figure 38-29. Burst write operation**




The RW split module accepts AXI commands, and transfers them to single read / write accesses on the SDRAM memory according to the ratio of the data width between the AXI bus and the SDRAM memory interface.

Inside the RW split module, there are two write FIFOs, which buffers the data and address of the AXI write commands. When neither of the write FIFOs is empty, write access occurs.

When the BRSTRD bit of EXMC_SDCTL0 register is set, the RW split module can anticipate the next read access. The read FIFOs are used to store data read in advance during the CAS latency period (configured by the CL bits of EXMC_SDCTLx) and during the PIPED delay (configured by the PIPED bits of EXMC_SDCTL0).

The RDATA FIFO can buffers up to 6 32-bit read data words, while the RADDR FIFO carries 6 14-bit read address tags to identify each of them. Every address tag is comprised of 11 bits for the column address, 2 bits for the internal bank address and 1 bit to select the SDRAM memories.

When there is a read commands on the AXI bus, the RW split module will firstly checks whether the address matches one of the address tags, and data are directly read from the FIFO when it is true. Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.

***Figure 38-30. Read access when FIFO not hit (BRSTRD=1, CL=2, SDCLK=2, PIPED=2)*** and ***Figure 38-31. Read access when FIFO hit (BRSTRD=1)*** specify the Read FIFO operation.

**Figure 38-30. Read access when FIFO not hit (BRSTRD=1, CL=2, SDCLK=2, PIPED=2)**

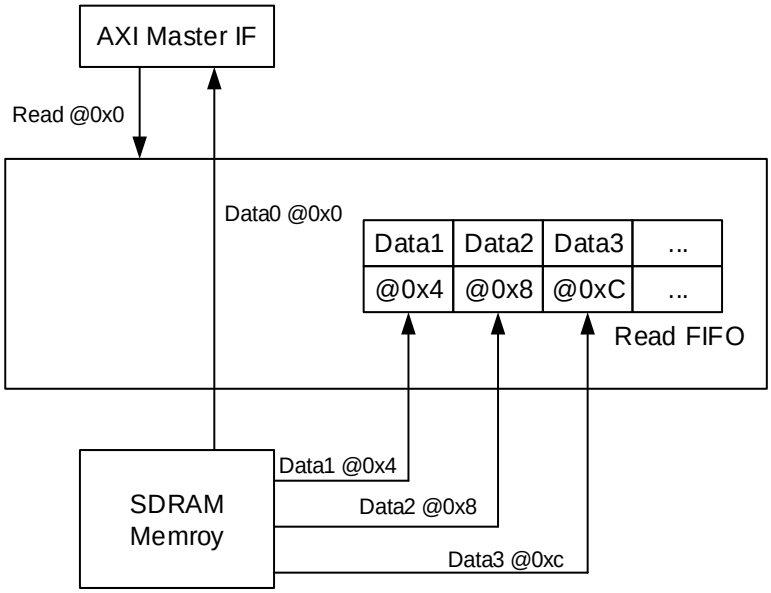
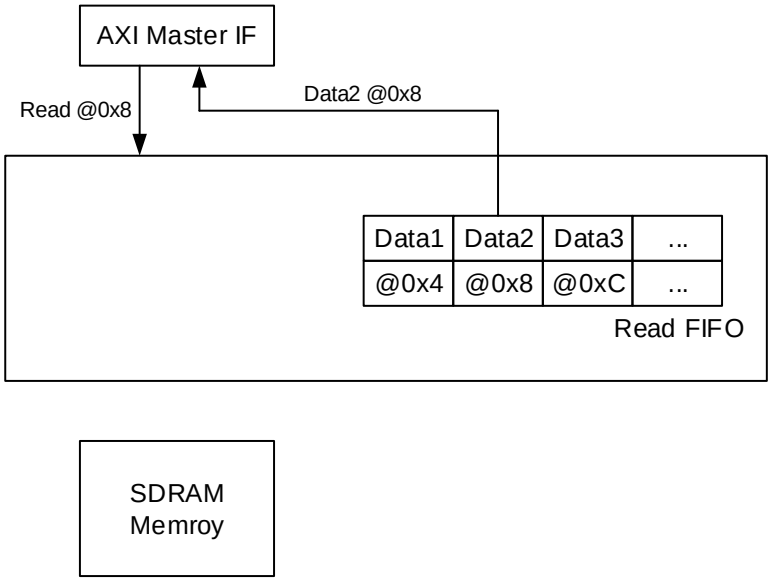



**Figure 38-31. Read access when FIFO hit (BRSTRD=1)**




The read FIFO will be flushed and ready to be filled with new data, when a write access or a precharge command occurs.

The address decoder sub-module translates the address of the AXI bus address to chip select, internal bank address, row address and column address according to the configuration of external memory device.

The active cache sub-module records whether the internal banks (up to 8) are in the active state. When an internal bank is in active state, the corresponding row address is also recorded. When an AXI access or an auto-refresh command is issued, the RW split module will look up this record and decide whether to generate the Active/Precharge commands or not.

Before read / write operation, the targeted row must be activated, the value of EXMC_A[15:14] selects the bank, and EXMC_A[12:0] select the row. The selected row remains active until a

precharge command is issued. The precharge command is used to deactivate an active row in a particular bank or the active row in all banks. A precharge command must be issued before activating a different row in the same bank. Active and precharge are automatically issued by the EXMC, its correctness depends on memory dimension configurations discussed previously, read and write timing diagram concerning automatic row activation and precharge are depicted as follows.

**Figure 38-32. Cross boundary read operation**

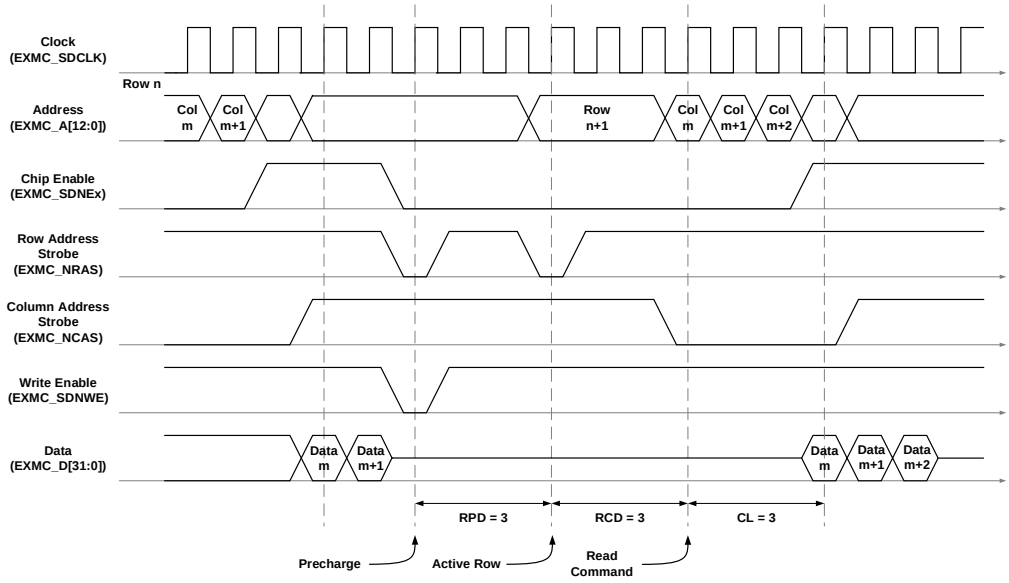
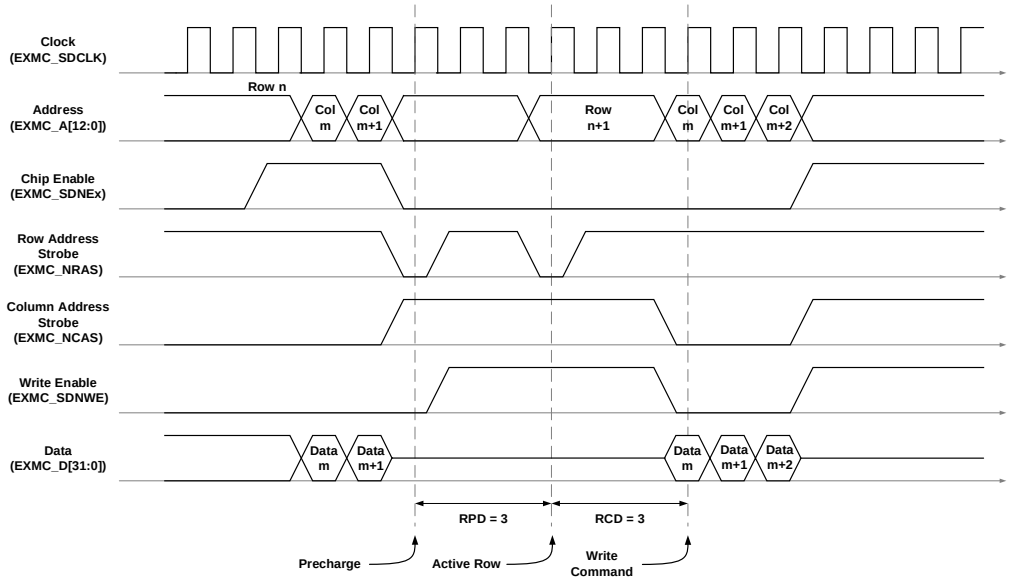



**Figure 38-33. Cross boundary write operation**




The above diagrams depict read and write timing waveform when memory access crosses row boundary, the following steps are preformed automatically:

1. Precharge the current active row.
2. Next row's activation.
3. Read / write access.

Precharge delay (PRD) and row to column delay (RCD) are added according to their

configuration in EXMC_SDTCFGx register, other timing parameters should be configured as SDRAM specification requires.

When this boundary happens to be at the end of a bank, two cases are possible:

1. When the current bank is not the last bank, the activation of the first row of the next bank is performed, and this supports all row, column, and bus width configuration.
2. When the current bank is the last bank, and row, column, and bus width are configured as, 13-bit, 11-bit, and 32-bit respectively, EXMC continues to read / write from the second SDRAM device (SDRAM device 1), assuming that the current SDRAM is device 0.

### Low power modes

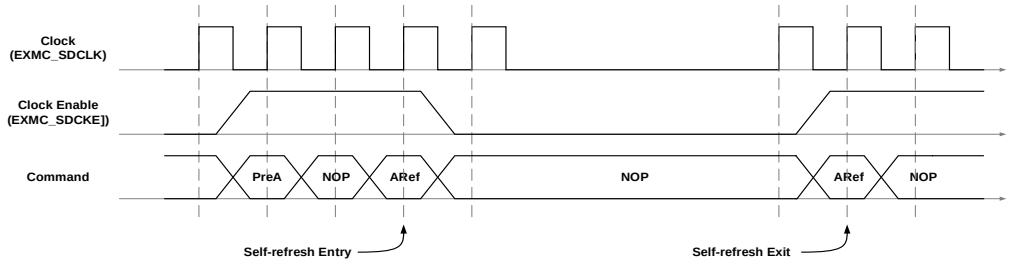
Two low power mode are supported:

1. Self-refresh mode: In self-refresh mode, refresh is provided by the SDRAM itself to maintain data integrity without external clock (EXMC_SDCLK). It is entered by writing 0b101 to CMD bits in EXMC_SDCMD register, DS0 andDS1 determines which SDRAM device will receive the command. EXMC_SDCLK stops running after a RASD delay if this command is issued to both SDRAM devices or one of the SDRAM device is not initialized.
2. Power-down mode: In power-down mode, refresh is provided by the SDRAM controller. It is entered by writing 0b110 to CMD bits in EXMC_SDCMD register, DS0 and DS1 determines which SDRAM device will receive the command. If the write data FIFO is not empty, all data are sent to the memory before activating power-down mode.

The Command Mode FSM also controls the switching process of between the normal mode and the low-power modes (self-refresh/power-down).

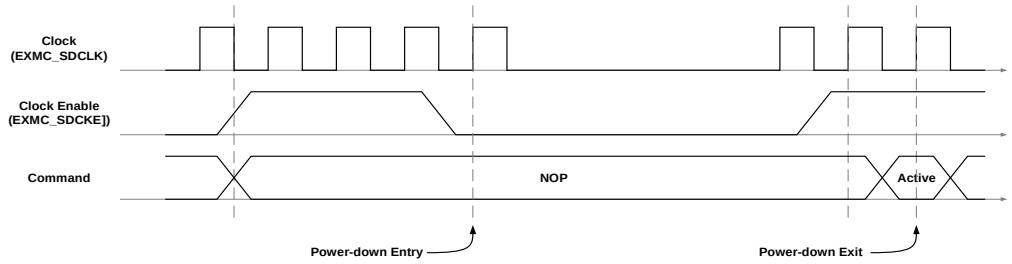
The SDRAM controller returns to normal mode from self-refresh mode when a read / write access occurs. If a read / write access occurs while the SDRAM controller is entering self-refresh mode, the self-refresh entry process will be interrupted, and the SDRAM controller remains in normal mode after the read / write access completed.

**Figure 38-34. Process for self-refresh entry and exit**




If an auto-refresh request occurs when the SDRAM controller is in power-down mode, the SDRAM controller returns to normal mode, issues the Precharge all and Auto-Refresh command sequence, and enters power-down mode again automatically.

**Figure 38-35. Process for power-down entry and exit**




## Status and interrupt

The not ready status NRDY bit in EXMC_SDSTAT register specifies whether the SDRAM controller is ready for a new command, this bit is cleared immediately after the command in the SDRAMC's internal register is sent.

Device0 and Device1 status bits STA0 and STA1 in EXMC_SDSTAT register defines the status of SDRAM deivce0 and device1 respectively, 0b00 represents normal mode, 0b01 indicates that the corresponding SDRAM devices is in self-refresh mode, and 0b10 signifies the power-down mode.

If a new refresh request occurs while the previous refresh command has not been served yet, a refresh error flag (REIF) is raised in EXMC_SDSTAT register, and interrupt is generated if REIE is set, refresh error flag is cleared by setting REC bit in EXMC_SDARI register.

## 38.4. Register definition

EXMC base address: 0x5200 4000

### 38.4.1. NOR/PSRAM controller registers

The peripheral registers have to be accessed by words (32-bit).

#### SRAM/NOR flash control registers (EXMC_SNCTLx) (x=0, 1, 2, 3)

Address offset: 0x00 + 8 * x, (x = 0, 1, 2, and 3)
Reset value: 0x0000 30DA

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | BKREMAP[1:0] | | Reserved | | | CCK | SYNCWR | CPS[2:0] | | |
| | | | | | | rw | | | | | rw | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASYNCWTEN | EXMODEN | NRWTEN | WEN | NRWTCFG | Reserved | NRWTPOL | SBRSTEN | Reserved | NREN | NRW[1:0] | | NRTP[1:0] | | NRMUX | NRBKEN |
| rw | rw | rw | rw | rw | | rw | rw | | rw | rw | | rw | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25:24 | BKREMAP[1:0] | Bank remap<br>00: Default mapping<br>01: NOR/PSRAM bank and SDRAM device 0 swapped<br>10: Reserved<br>11: Reserved<br>**Note:** The BKREMAP bits are only present in EXMC_SNCTLR0 register, and thess bits position in EXMC_SNCTLx (x = 1, 2, 3) registers are mearningless. |
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | CCK | Consecutive clock<br>0: EXMC_CLK is generated only during synchronous access.<br>1: EXMC_CLK is generated unconditionally.<br>**Note**: Consecutive clock (CCK) bit is only present in EXMC_SNCTLR0 register, and this bit position in EXMC_SNCTLx (x = 1, 2, 3) registers is mearningless.<br>When this bit is set, only CKDIV[3:0] bits in EXMC_SNTCFG0 can effect EXMC_CLK output. |
| 19 | SYNCWR | Synchronous write<br>0: Asynchronous write<br>1: Synchronous write |

| | | |
|---|---|---|
| 18:16 | CPS[2:0] | CRAM page size<br>000: Automatic burst split on page boundary crossing<br>001: 128 bytes<br>010: 256 bytes<br>011: 512 bytes<br>100: 1024 bytes<br>Others: Reserved |
| 15 | ASYNCWTEN | Asynchronous wait enable<br>0: Disable the asynchronous wait function<br>1: Enable the asynchronous wait function |
| 14 | EXMODEN | Extended mode enable<br>0: Disable extended mode<br>1: Enable extended mode |
| 13 | NRWTEN | NWAIT signal enable<br>For flash memory access in burst mode, this bit enables/disables wait-state insertion to the NWAIT signal.<br>0: Disable NWAIT signal<br>1: Enable NWAIT signal |
| 12 | WEN | Write enable<br>0: Disable write in the bank by the EXMC, otherwise an AXI error is reported<br>1: Enable write in the bank by the EXMC (default after reset) |
| 11 | NRWTCFG | NWAIT signal configuration, only work in synchronous mode<br>0: NWAIT signal is active one data cycle before wait state<br>1: NWAIT signal is active during wait state |
| 10 | Reserved | Must be kept at reset value. |
| 9 | NRWTPOL | NWAIT signal polarity<br>0: Low level of NWAIT is active<br>1: High level of NWAIT is active |
| 8 | SBRSTEN | Synchronous burst enable<br>0: Disable burst access mode<br>1: Enable burst access mode |
| 7 | Reserved | Must be kept at reset value. |
| 6 | NREN | NOR Flash access enable<br>0: Disable NOR Flash access<br>1: Enable NOR Flash access |
| 5:4 | NRW[1:0] | NOR region memory data bus width<br>00: 8 bits<br>01: 16 bits(default after reset)<br>10: 32 bits |

| | | |
|---|---|---|
| | | 11: Reserved |
| 3:2 | NRTP[1:0] | NOR region memory type<br>00: SRAM, ROM<br>01: PSRAM (CRAM)<br>10: NOR Flash<br>11: Reserved |
| 1 | NRMUX | NOR region memory address/data multiplexing<br>0: Disable address/data multiplexing function<br>1: Enable address/data multiplexing function |
| 0 | NRBKEN | NOR region enable<br>0: Disable the corresponding memory bank<br>1: Enable the corresponding memory bank |

### SRAM/NOR flash timing configuration registers (EXMC_SNTCFGx) (x=0, 1, 2, 3)

Address offset: 0x04 + 8 * x, (x = 0, 1, 2, and 3)

Reset value: 0x0FFF FFFF

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | ASYNCMOD[1:0] | | DLAT[3:0] | | | | CKDIV[3:0] | | | | BUSLAT[3:0] | | | |
| | | rw | | rw | | | | rw | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DSET[7:0] | | | | | | | | AHLD[3:0] | | | | ASET[3:0] | | | |
| rw | | | | | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:28 | ASYNCMOD[1:0] | Asynchronous access mode<br>The bits are valid only when the EXMODEN bit in the EXMC_SNCTLx register is 1.<br>00: Mode A access<br>01: Mode B access<br>10: Mode C access<br>11: Mode D access |
| 27:24 | DLAT[3:0] | Data latency for NOR Flash. Only valid in synchronous access<br>0x0: Data latency of first burst access is 2 CLK<br>0x1: Data latency of first burst access is 3 CLK<br>……<br>0xF: Data latency of first burst access is 17 CLK |
| 23:20 | CKDIV[3:0] | Synchronous clock divide ratio. This filed is only effect in synchronous mode.<br>0x0: No EXMC_CLK output |

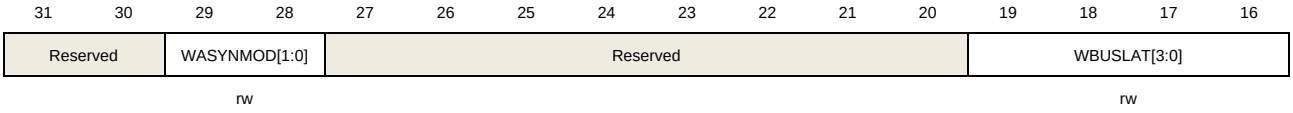
| | | |
|---|---|---|
| | | 0x1: EXMC_CLK period = 2 * CK_EXMC period<br>……<br>0xF: EXMC_CLK period = 16 * CK_EXMC period |
| 19:16 | BUSLAT[3:0] | Bus latency<br>The bits are defined in multiplexed read mode in order to avoid bus contention, and the bits represent the minimum time the data bus used to return to a high impedance state.<br>0x0: Bus latency = 0 * CK_EXMC period<br>0x1: Bus latency = 1 * CK_EXMC period<br>……<br>0xF: Bus latency = 15 * CK_EXMC period |
| 15:8 | DSET[7:0] | Data setup time<br>This field is meaningful only in asynchronous access.<br>0x00: Reserved<br>0x01: Data setup time = 1 * CK_EXMC period<br>……<br>0xFF: Data setup time = 255 * CK_EXMC period |
| 7:4 | AHLD[3:0] | Address hold time<br>This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode.<br>0x0: Reserved<br>0x1: Address hold time = 1 * CK_EXMC<br>……<br>0xF: Address hold time = 15 * CK_EXMC |
| 3:0 | ASET[3:0] | Address setup time<br>This field is used to set the time of address setup phase.<br>**Note:** meaningful only in asynchronous access of SRAM,ROM,NOR Flash<br>0x0: Address setup time = 0 * CK_EXMC<br>……<br>0xF: Address setup time = 15 * CK_EXMC |

### SRAM/NOR flash write timing configuration registers (EXMC_SNWTCFGx) (x=0, 1, 2, 3)

Address offset: 0x104 + 8 * x, (x = 0, 1, 2, and 3)
Reset value: 0x0FFF FFFF

This register is meaningful only when the EXMODEN bit in EXMC_SNCTLx is set to 1.

This register has to be accessed by word(32-bit)



| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDSET[7:0] | | | | | | | | WAHLD[3:0] | | | | WASET[3:0] | | | |
| rw | | | | | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:28 | WASYNCMOD[1:0] | Asynchronous access mode<br>The bits are valid only when the EXMODEN bit in the EXMC_SNCTLx register is 1.<br>00: Mode A access<br>01: Mode B access<br>10: Mode C access<br>11: Mode D access |
| 27:20 | Reserved | Must be kept at reset value. |
| 19:16 | WBUSLAT[3:0] | Bus latency<br>Bus latency added at the end of each write transaction to match with the minimum time between consecutive transactions.<br>0x0: Bus latency = 0 * CK_EXMC period<br>0x1: Bus latency = 1 * CK_EXMC period<br>……<br>0xF: Bus latency = 15 * CK_EXMC period |
| 15:8 | WDSET[7:0] | Data setup time<br>This field is meaningful only in asynchronous access.<br>0x00: Reserved<br>0x01: Data setup time = 1 * CK_EXMC period<br>……<br>0xFF: Data setup time = 255 * CK_EXMC period |
| 7:4 | WAHLD[3:0] | Address hold time<br>This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode.<br>0x0: Reserved<br>0x1: Address hold time = 1 * CK_EXMC<br>……<br>0xF: Address hold time = 15 * CK_EXMC |
| 3:0 | WASET[3:0] | Address setup time<br>This field is used to set the time of address setup phase.<br>**Note:** Meaningful only in asynchronous access of SRAM,ROM,NOR Flash<br>0x0: Address setup time = 0 * CK_EXMC<br>0x1: Address setup time = 1 * CK_EXMC<br>……<br>0xF: Address setup time = 15 * CK_EXMC |

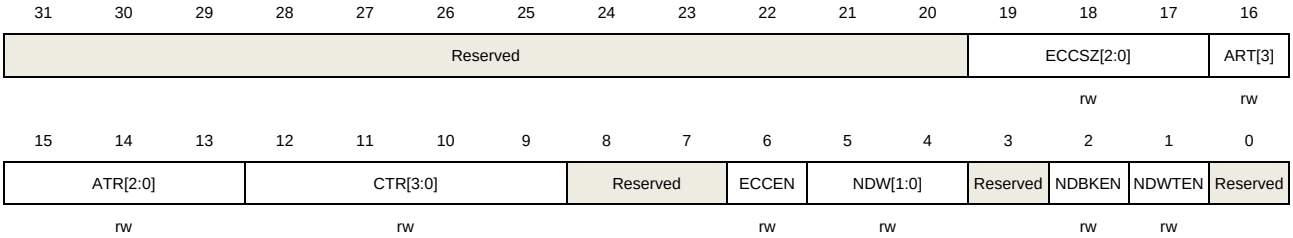
## 38.4.2. NAND flash controller registers

### NAND flash control registers (EXMC_NCTL)

Address offset: 0x80

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value. |
| 19:17 | ECCSZ[2:0] | ECC size<br>000: 256 bytes<br>001: 512 bytes<br>010: 1024 bytes<br>011: 2048 bytes<br>100: 4096 bytes<br>101: 8192 bytes |
| 16:13 | ATR[3:0] | ALE to RE delay<br>0x0: ALE to RE delay = 1 * CK_EXMC<br>……<br>0xF: ALE to RE delay = 16 * CK_EXMC |
| 12:9 | CTR[3:0] | CLE to RE delay<br>0x0: CLE to RE delay = 1 * CK_EXMC<br>0x1: CLE to RE delay = 2 * CK_EXMC<br>……<br>0xF: CLE to RE delay = 16 * CK_EXMC |
| 8:7 | Reserved | Must be kept at reset value. |
| 6 | ECCEN | ECC enable<br>0: Disable ECC, and reset EXMC_NECC<br>1: Enable ECC |
| 5:4 | NDW[1:0] | NAND bank memory data bus width<br>00: 8 bits<br>01: 16 bits<br>Others: Reserved |

| | | |
|---|---|---|
| 3 | Reserved | Must be kept at reset value. |
| 2 | NDBKEN | NAND bank enable<br>0: Disable corresponding memory bank<br>1: Enable corresponding memory bank |
| 1 | NDWTEN | Wait function enable<br>0: Disable wait function<br>1: Enable wait function |
| 0 | Reserved | Must be kept at reset value. |

### NAND flash interrupt enable registers (EXMC_NINTEN)

Address offset: 0x84

Reset value: 0x0000 0042

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | FFEPT | INTFEN | INTHEN | INTREN | INTFS | INTHS | INTRS |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | FFEPT | FIFO empty flag<br>0: FIFO is not empty.<br>1: FIFO is empty. |
| 5 | INTFEN | Interrupt falling edge detection enable<br>0: Disable interrupt falling edge detection<br>1: Enable interrupt falling edge detection |
| 4 | INTHEN | Interrupt high-level detection enable<br>0: Disable interrupt high-level detection<br>1: Enable interrupt high-level detection |
| 3 | INTREN | Interrupt rising edge detection enable bit<br>0: Disable interrupt rising edge detection<br>1: Enable interrupt rising edge detection |
| 2 | INTFS | Interrupt falling edge status<br>0: Not detect interrupt falling edge<br>1: Detect interrupt falling edge |
| 1 | INTHS | Interrupt high-level status |

| | | |
|---|---|---|
| | | 0: Not detect interrupt high-level<br>1: Detect interrupt high-level |
| 0 | INTRS | Interrupt rising edge status<br>0: Not detect interrupt rising edge<br>1: Detect interrupt rising edge |

### NAND flash common space timing configuration registers (EXMC_NCTCFG)

Address offset: 0x88

Reset value: 0xFFFF FFFF

These operations applicable to common memory space for NAND Flash.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMHIZ[7:0] | | | | | | | | COMHLD[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COMWAIT[7:0] | | | | | | | | COMSET[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | COMHIZ[7:0] | Common memory data bus HIZ time<br>The bits are defined as time of bus keep high impedance state after writing the data.<br>0x00: COMHIZ = 1 * CK_EXMC<br>……<br>0xFE: COMHIZ = 255 * CK_EXMC<br>0xFF: Reserved |
| 23:16 | COMHLD[7:0] | Common memory hold time<br>After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time.<br>0x00: Reserved<br>0x01: COMHLD = 1 * CK_EXMC<br>……<br>0xFE: COMHLD = 254 * CK_EXMC<br>0xFF: Reserved |
| 15:8 | COMWAIT[7:0] | Common memory wait time<br>Define the minimum time to maintain command<br>0x00: Reserved<br>0x01: COMWAIT = 2 * CK_EXMC (+NWAIT active cycles)<br>……<br>0xFE: COMWAIT = 255 * CK_EXMC (+NWAIT active cycles) |

| | | |
|---|---|---|
| | | 0xFF: Reserved |
| 7:0 | COMSET[7:0] | Common memory setup time<br>Define the time to build address before sending command<br>0x00: COMSET = 1 * CK_EXMC<br>……<br>0xFE: COMSET = 255 * CK_EXMC<br>0xFF: Reserved |

### NAND flash attribute space timing configuration registers (EXMC_NATCFG)

Address offset: 0x8C

Reset value: 0xFFFF FFFF

It is used for 8-bit accesses to the attribute memory space of the NAND Flash for the last address write access if another timing must be applied.

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ATTHIZ[7:0] | | | | | | | | ATTHLD[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ATTWAIT[7:0] | | | | | | | | ATTSET[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | ATTHIZ[7:0] | Attribute memory data bus HIZ time<br>The bits are defined as time of bus keep high impedance state after writing the data.<br>0x00: ATTHIZ = 0 * CK_EXMC<br>…...<br>0xFE: ATTHIZ = 254 * CK_EXMC<br>0xFF: Reserved |
| 23:16 | ATTHLD[7:0] | Attribute memory hold time<br>After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time.<br>0x00: Reserved<br>0x01: ATTHLD = 1 * CK_EXMC<br>……<br>0xFE: ATTHLD = 254 * CK_EXMC<br>0xFF: Reserved |
| 15:8 | ATTWAIT[7:0] | Attribute memory wait time<br>Define the minimum time to maintain command<br>0x00: Reserved<br>0x01: ATTWAIT = 2 * CK_EXMC (+NWAIT active cycles) |

……

0xFE: ATTWAIT = 255 * CK_EXMC (+NWAIT active cycles)

0xFF: ATTWAIT = Reserved

| | | |
|---|---|---|
| 7:0 | ATTSET[7:0] | Attribute memory setup time<br>Define the time to build address before sending command<br>0x00: ATTSET = 1 * CK_EXMC<br>……<br>0xFE: ATTSET = 255 * CK_EXMC<br>0xFF: Reserved |

#### NAND flash ECC registers (EXMC_NECC)

Address offset: 0x94

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECC[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ECC[31:0] | ECC result |

| ECCSZ[2:0] | NAND Flash page size | ECC bits |
|---|---|---|
| 0b000 | 256 | ECC[21:0] |
| 0b001 | 512 | ECC[23:0] |
| 0b010 | 1024 | ECC[25:0] |
| 0b011 | 2048 | ECC[27:0] |
| 0b100 | 4096 | ECC[29:0] |
| 0b101 | 8192 | ECC[31:0] |

### 38.4.3. SDRAM controller registers

#### SDRAM control registers (EXMC_SDCTLx) (x=0, 1)

Address offset: 0x140+4*x, (x = 0, 1)

Reset value: 0x0000 02D0

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 13 | 12 | 11 10 | 9 | 8 7 | 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|---|---|
| SDCLK[2] | PIPED[1:0] | BRSTRD | SDCLK[1:0] | WPEN | CL[1:0] | NBK | SDW[1:0] | RAW[1:0] | CAW[1:0] |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Forced by hardware to 0. |
| 15 | SDCLK[2] | Refer to SDCLK[1:0] description. |
| 14:13 | PIPED[1:0] | Pipeline delay<br>These bits specify the delay for reading data after CAS latency in CK_EXMC clock cycles.<br>00: 0 CK_EXMC clock cycle delay<br>01: 1 CK_EXMC clock cycle delay<br>10: 2 CK_EXMC clock cycle delay<br>11: reserved<br>**Note:** The corresponding bits in the EXMC_SDCTL1 register are reserved. |
| 12 | BRSTRD | Burst read<br>When this bit is set, The SDRAM controller anticipates the next read commands during the CAS latency and stores data in the Read FIFO.<br>0: burst read disabled<br>1: burst read enabled<br>**Note:** The corresponding bits in the EXMC_SDCTL1 register are reserved. |
| 11:10 | SDCLK[1:0] | SDRAM clock configuration<br>These bits specifies the SDRAM clock period for both SDRAM devices. The memory clock should be disabled before change, and the SDRAM memory must be re-initialized after this configuration is changed.<br>000: SDCLK memory clock disabled<br>001: Reserved<br>010: SDCLK memory period = 2 x CK_EXMC periods<br>011: SDCLK memory period = 3 x CK_EXMC periods<br>110: SDCLK memory period = 4 x CK_EXMC periods<br>111: SDCLK memory period = 5 x CK_EXMC periods<br>Others: Reserved<br>**Note:** The corresponding bits in the EXMC_SDCTL1 register are reserved.<br>The SDCLK[2] bit is not contiguous, which is located in bit 15. |
| 9 | WPEN | Write protection enable<br>This bit enables the write protection function.<br>0: Disable write protection, write accesses allowed<br>1: Enable write protection, write accesses ignored |
| 8:7 | CL[1:0] | CAS Latency<br>This bits sets specifies SDRAM CAS latency in SDRAM memory clock cycle unit |

| | | |
|---|---|---|
| | | 00: reserved, do not use.<br>01: 1 cycle<br>10: 2 cycles<br>11: 3 cycles |
| 6 | NBK | Number of banks<br>This bit specifies the number of internal banks.<br>0: 2 internal Banks<br>1: 4 internal Banks |
| 5:4 | SDW[1:0] | SDRAM data bus width.<br>These bits specify the SDRAM memory data width.<br>00: 8 bits<br>01: 16 bits<br>10: 32 bits<br>11: reserved |
| 3:2 | RAW[1:0] | Row address bit width<br>These bits specify the bit width of a row address.<br>00: 11 bit<br>01: 12 bits<br>10: 13 bits<br>11: reserved |
| 1:0 | CAW[1:0] | Column address bit width<br>These bits specify the bit width of column address.<br>00: 8 bits<br>01: 9 bits<br>10: 10 bits<br>11: 11 bits. |

### SDRAM timing configuration registers (EXMC_SDTCFGx) (x=0, 1)

Address offset: 0x148+4*x, (x = 0, 1)

Reset value: 0x0FFF FFFF

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | RCD[3:0] | | | | RPD[3:0] | | | | WRD[3:0] | | | |
| | | | | rw | | | | rw | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARFD[3:0] | | | | RASD[3:0] | | | | XSRD[3:0] | | | | LMRD[3:0] | | | |
| rw | | | | rw | | | | rw | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Forced by hardware to 0. |

| Bits | Fields | Descriptions |
|---|---|---|
| 27:24 | RCD[3:0] | Row to column delay<br>These bits specify the delay between an Activate command and a Read / write command in SDRAM memory clock cycle unit.<br>0x0: 1 cycle.<br>0x1: 2 cycles<br>....<br>0xF: 16 cycles |
| 23:20 | RPD[3:0] | Row precharge delay<br>These bits specify the delay between a Precharge command and the next command in SDRAM memory clock cycle unit.<br>0x0: 1 cycle<br>0x1: 2 cycles<br>....<br>0xF: 16 cycles<br>**Note:** The corresponding bits in the EXMC_SDTCFG1 register are reserved. If two SDRAM memories are used, the RPD must be programmed with the timings of the slower one. |
| 19:16 | WRD[3:0] | Write recovery delay<br>These bits specify the delay between a Write and a Precharge command in SDRAM memory clock cycle unit.<br>0x0: 1 cycle<br>0x1: 2 cycles<br>.....<br>0xF: 16 cycles<br>**Note:** The corresponding bits in the EXMC_SDTCFG1 register are reserved. If two SDRAM memories are used, the WRD must be programmed with the timings of the slower one. |
| 15:12 | ARFD[3:0] | Auto refresh delay<br>These bits specify the delay between two consecutive Refresh commands, the delay between two Activate commands, as well as the delay between the Refresh command and the Activate command in SDRAM memory clock cycle unit.<br>0x0: 1 cycle<br>0x1: 2 cycles<br>....<br>0xF: 16 cycles<br>**Note:** The corresponding bits in the EXMC_SDTCFG1 register are reserved. If two SDRAM memories are used, the ARFD must be programmed with the timings of the slower one. |
| 11:8 | RASD[3:0] | Row address select delay |

| Bits | Fields | Descriptions |
|---|---|---|
| | | These bits specify the delay between an Activate command and a Precharge command in SDRAM memory clock cycle unit. The minimum delay between two successive Self-refresh commands is also specified by these bits.<br>0x0: 1 cycle<br>0x1: 2 cycles<br>.....<br>0xF: 16 cycles |
| 7:4 | XSRD[3:0] | Exit Self-refresh delay<br>These bits specify the delay from a Self-refresh command to an Activate command in SDRAM memory clock cycle unit.<br>0x0: 1 cycle<br>0x1: 2 cycles<br>……<br>0xF: 16 cycles |
| 3:0 | LMRD[3:0] | Load Mode Register Delay<br>These bits specify the delay between a Load Mode Register command and a Refresh or Active command in SDRAM memory clock cycle unit.<br>0x0: 1 cycle<br>0x1: 2 cycles<br>.....<br>0xF: 16 cycles |

### SDRAM command register (EXMC_SDCMD)

Address offset: 0x150

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | MRC[12:7] | | | | | |
| | | | | | | | | | | rw | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MRC[6:0] | | | | | | | NARF[3:0] | | | | DS0 | DS1 | CMD[2:0] | | |
| rw | | | | | | | rw | | | | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Forced by hardware to 0. |
| 21:9 | MRC[12:0] | Mode register content<br>These bits specify the SDRAM Mode Register content which will be programmed when CMD = ‘100’. |
| 8:5 | NARF[3:0] | Number of successive Auto-refresh |

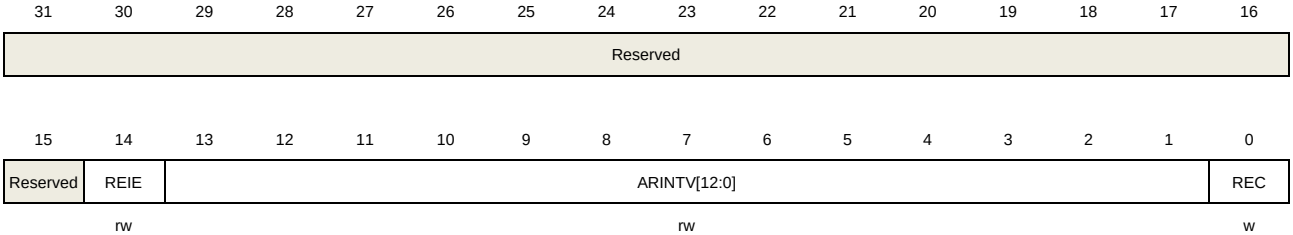
| | | |
|---|---|---|
| | | These bits specify how many successive Auto-refresh cycles will be send when CMD = '011'.<br>0x0: 1 Auto-refresh cycle<br>0x1: 2 Auto-refresh cycles<br>....<br>0xE: 15 Auto-refresh cycles<br>0xF: Reserved |
| 4 | DS0 | Device select 0<br>This bit indicates whether the SDRAM Device0 is selected or not.<br>0: SDRAM Device0 is not selected<br>1: SDRAM Device0 is selected |
| 3 | DS1 | Device select 1<br>This bit indicates whether the SDRAM Device1 is selected or not.<br>0: SDRAM Device1 is not selected<br>1: SDRAM Device1 is selected |
| 2:0 | CMD[2:0] | Command<br>These bits specify the commands, which are issued to the SDRAM device.<br>000: Normal operation command<br>001: Clock enable command<br>010: Precharge All command<br>011: Auto-refresh command<br>100: Load Mode Register command<br>101: Self-refresh command<br>110: Power-down entry command<br>111: Reserved<br>**Note:** At least one command device select bit (DS1 or DS0) must be set, when a command is issued. If both devices are used, the commands must be issued to the two devices by setting the DS1and DS0 bits at the same time. |

### SDRAM auto-refresh interval register (EXMC_SDARI)

Address offset: 0x154

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|

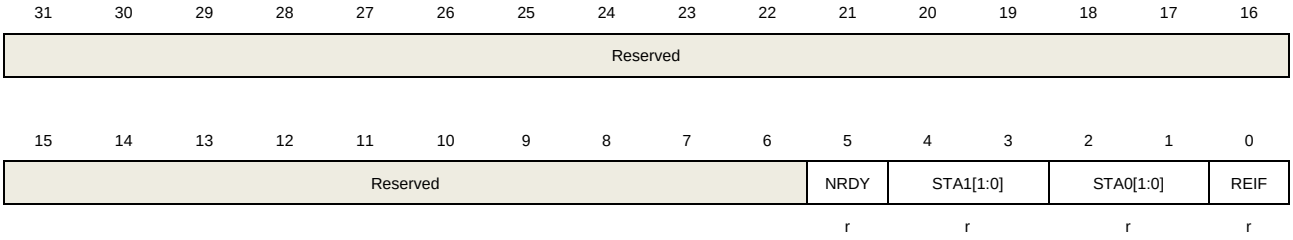
| | | |
|---|---|---|
| 31:15 | Reserved | Forced by hardware to 0. |
| 14 | REIE | Refresh error interrupt Enable<br>0: Interrupt is disabled<br>1: An Interrupt is generated if REIF bit of the Status Register is set |
| 13:1 | ARINTV[12:0] | Auto-Refresh Interval<br>This bit field specifies the interval of two successive auto-refresh commands in memory clock cycle unit.<br>ARFITV = (SDRAM refresh period / Number of rows) - 20 |
| 0 | REC | Refresh error flag clear<br>The Refresh Error Flag (REIF) in the Status Register will be cleared when this bit is set.<br>0: no effect<br>1: Clear the Refresh Error flag |

### SDRAM status register (EXMC_SDSTAT)

Address offset: 0x158

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Forced by hardware to 0. |
| 5 | NRDY | Not Ready status<br>This bit specifies whether the SDRAM controller is ready for a new command<br>0: SDRAM Controller is ready for a new command<br>1: SDRAM Controller is not ready for a new command |
| 4:3 | STA1[1:0] | Device1 status<br>This bit defines the Status of SDRAM Device1.<br>00: Normal status<br>01: Self-refresh status<br>10: Power-down status |
| 2:1 | STA0[1:0] | Device 0 status<br>This bit defines the Status of SDRAM Device 0.<br>00: Normal status<br>01: Self-refresh status |

| | | |
|---|---|---|
| | | 10: Power-down status |
| 0 | REIF | Refresh error interrupt flag<br>0: No refresh error<br>1: A refresh error occurred. An interrupt is generated when REIE = 1. |

### SDRAM read sample control register (EXMC_SDRSCTL)

Address offset: 0x180

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | SDSC[3:0] | | | | Reserved | | SSCR | RSEN |
| | | | | | | | | rw | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Forced by hardware to 0. |
| 7:4 | SDSC[3:0] | Select the delayed sample clock of read data<br>0x0: Select the clock after 0 delay cell<br>0x1: Select the clock after 1 delay cell<br>……<br>0xF: Select the clock after 15 delay cell |
| 3:2 | Reserved | Forced by hardware to 0. |
| 1 | SSCR | Select sample cycle of read data<br>0: add 0 extra CK_EXMC cycle to the read data sample clock besides the delay chain<br>1: add 1 extra CK_EXMC cycle to the read data sample clock besides the delay chain |
| 0 | RSEN | Read sample enable<br>0: Read sample disabled<br>1: Read sample enabled |

# 39. VREF

## 39.1. Overview

A precision internal reference circuit is inside. The internal voltage reference unit is used to provide voltage reference for ADC / DAC, or used by off-chip circuit connecting to VREFP pin.

## 39.2. Characteristics

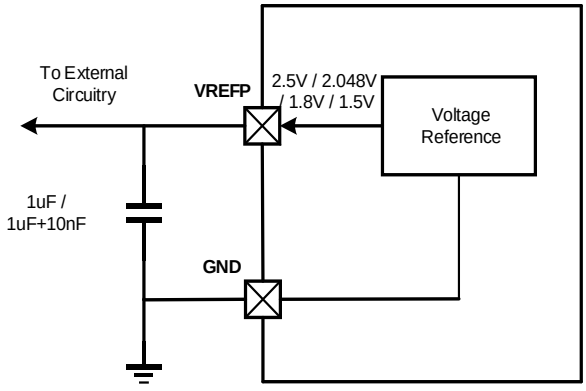
- Stable voltage, and product calibrated.
- Connects to VREFP pin to source off-chip circuits.
- 1.5V, 1.8V, 2.048V or 2.5V configurable reference voltage output.

## 39.3. Function overview

The VREF is enabled by set the VREFEN bit in VREF_CS register, and the VREF output can be configured to be either 1.5V, 1.8V, 2.048V, or 2.5V by programming the VREFS[1:0] bits. When VREF is enabled and the HIPM bit is reset, the internal voltage reference can be connected to VREFP pin. When VREF is disabled and the HIPM bit is set, off-chip voltage reference can be injected to VREFP pin to source ADC / DAC. If there is no VREFP pin (refer to datasheet), the VREFP is connected to VDDA and the VREFEN bit must keep 0.

When using precision internal voltage reference, and a bypass capacitor about 1uF (or 1uF and 10nF connected in parallel) which is recommended to ground is required.

**Figure 39-1.Precision reference connection**




As shown in ***Table 39-1. VREF modes***, the precision internal reference voltage unit can work in four kinds of mode by programming the VREFEN and HIPM bits in the VREF_CS register.

**Table 39-1. VREF modes**

| VREFEN | HIPM | Mode |
|---|---|---|
| 0 | 0 | VREF disabled, |

| VREFEN | HIPM | Mode |
|---|---|---|
| | | VREFP pin pulled-down to VSSA. |
| 0 | 1 | External voltage reference mode (default):<br>VREF disabled.<br>off-chip reference voltage injected from VREFP pin. |
| 1 | 0 | Internal voltage reference mode:<br>VREF enabled.<br>VREFP pin inside connected to VREF output. |
| 1 | 1 | Hold mode:<br>VREF disabled.<br>VREFP pin floating. The voltage is maintained by the external capacitor.<br>VREFRDY detection disabled and VREFRDY bit keeps last state. |

When VREF is configured in internal voltage reference mode by setting VREFEN bit and reset HIPM bit in the VREF_CS register, the user must wait before VREFRDY bit is set, indicating that the VREF output has attained the set value.

## 39.4. Register definition

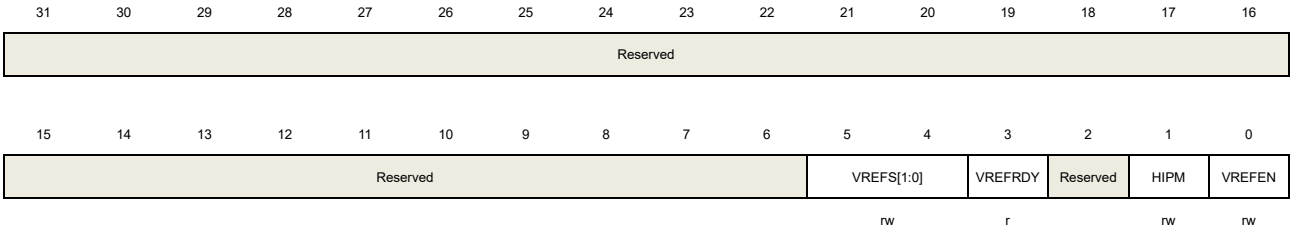
VREF base address: 0x5800 3C00

### 39.4.1. Control and status register (VREF_CS)

Address offset: 0x00

Reset value: 0x0000 0002

This register can be accessed by half-word (16-bit) or word (32-bit).



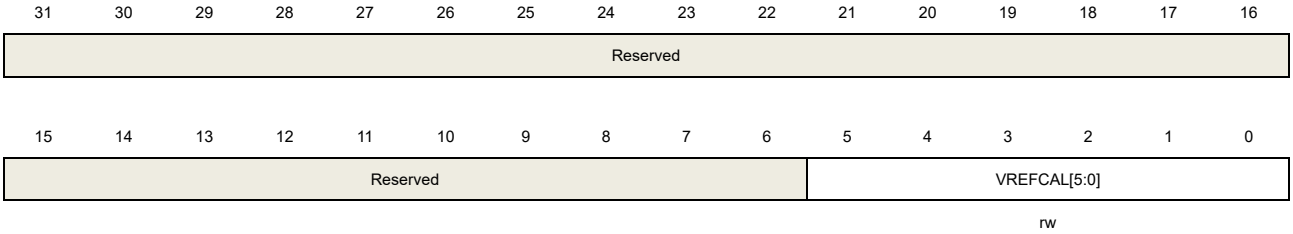

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:4 | VREFS[1:0] | Voltage reference select<br>These bits set the value of voltage reference output by the VREF.<br>00: The voltage reference is around 2.5 V<br>01: The voltage reference is around 2.048 V<br>10: The voltage reference is around 1.8 V<br>11: The voltage reference is around 1.5 V<br>This bit can be modified only when the VREF is disabled (VREFEN = 0). |
| 3 | VREFRDY | VREF ready<br>0: The output of the VREF does not attain the set value<br>1: The output of the VREF attains the set value |
| 2 | Reserved | Must be kept at reset value. |
| 1 | HIPM | High impedance state<br>0: The VREFP pin is inside connected to the VREF output<br>1: The VREFP pin is set to high impedance state |
| 0 | VREFEN | VREF enable<br>0: VREF is disabled<br>1: VREF is enabled |

### 39.4.2. Calibration register (VREF_CALIB)

Address offset: 0x04

Reset value: 0x0000 00XX

This register can be accessed by half-word (16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | VREFCAL[5:0] | VREF calibration<br>After reset, these bits will be initialized to the calibration value saved in the Flash memory during the course of production test. Writing to these bits can adjust the output of the VREF.<br>**Note:** If the user performs the calibration procedure, the VREF calibration must increase progressively from 0x00 to 0x3F. |

# 40. Low power digital temperature sensor (LPDTS)

## 40.1. Overview

Low power digital tempearature sensor(LPDTS) is used to transmit square wave,which is converted by temperature and the frequency is proportional to the absolute temperature. The frequency measurement is based on the PCLK or the LXTAL clock.

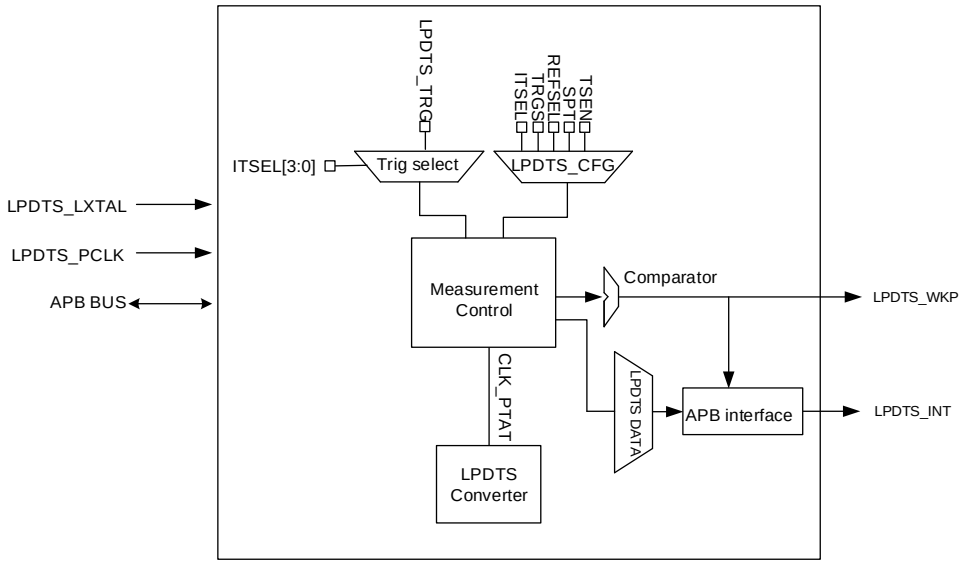
## 40.2. Characteristics

- The trigger source of measurement can be set to software or hardware.
- Programmable sampling time.
- Temperature window watchdog.
- The interrupt can be generated when the temperature is below a low threshold or above a high threshold and at the end of measurement.
- The generation of asynchronous wakeup signal indicates that the measurement result is higher or lower than the specified threshold when the LXTAL is selected as reference clock.

## 40.3. Block diagram

*Figure 40-1. LPDTS block diagram* shows the LPDTS block diagram.

**Figure 40-1. LPDTS block diagram**



## 40.4. Function overview

### 40.4.1. LPDTS internal signals

**Table 40-1. LPDTS signals**

| Signal name | Type | Description |
|---|---|---|
| LPDTS_LXTAL | input | LXTAL clock |
| LPDTS_PCLK | input | APB clock |
| LPDTS_INT | output | The interrupt of intenal temperature sensor |
| LPDTS_WKP | output | The wakeup of intenal temperature sensor |

### 40.4.2. Operating modes

The REFSEL bit in LPDTS_CFG can be set to selected multiple operation modes.

- PCLK mode (REFSEL = 0)

The register can be wrote or read by software. The REFSEL bit is set to 0 to select the PCLK as reference clock.

- PCLK and LXTAL mode (REFSEL = 1)

The register can be wrote or read by software. The REFSEL bit is set to 1 to select the LXTAL as reference clock.

- LXTAL mode (REFSEL = 1 and PCLK OFF)

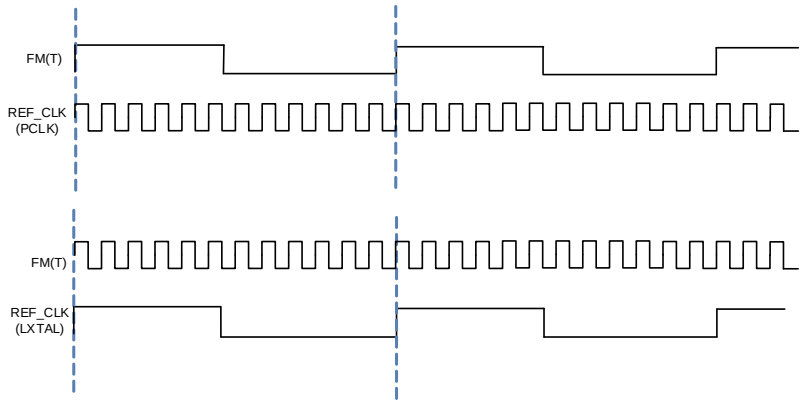
The temperature sensor registers cannot be accessed. The LXTAL is selected as reference clock. This mode can use hardware triggers to exit the Deep-sleep mode.

### 40.4.3. Temperature measurement principles

A signal is output, which FM(T) frequency (typically 641 kHz) is related to temperature, by the analog part of temperature sensor. Two counters are embedded in the temperature sensor block, which makes the counting mode relevant to the reference clock frequency. The counting result is stored in the LPDTS_DATA register.

- When the reference clock is PCLK, the measurement method is to sample one or multiple FM(T) cycles and count at the rising edge and falling edge of PCLK.
- When the reference clock is LXTAL, the measurement method is to sample one or multiple LXTAL cycles and count at the rising edge and falling edge of FM(T).

**Figure 40-2. Method for different REF_CLK**




The Temperature calculation formula When PCLK is used:

$$T=T0+((2\times F_{PCLK}/COVAL)\times SPT-100\times FREQ)/RF_CF \quad (40\text{-}1)$$

The Temperature calculation formula When LXTAL is used:

$$T=T0+(((F_{LXTAL}\times COVAL)/(2\times SPT))-(100\times FREQ))/RF_CF \quad (40\text{-}2)$$

where:

- T0 is equal to 25 °C.
- COVAL is the value of the counter output value for temperature sensor which measured and stored in the LPDTS_DATA register.
- SPT is Sampling time for temperature sensor
- FREQ is engineering value of the frequency measured at T0 for temperature sensor which measured and stored in the LPDTS_SDATA register. It is expressed in hundreds of Hertz.
- RF_CF is the engineering value of the ramp coefficient for the temperature sensor.

### 40.4.4. Sampling time

Increasing the sampling period helps to improve the measurement accuracy. It is most effective when the reference frequency is set close to the sampling frequency.

The default value of the sampling time should be set as one REF_CLK cycle or one FM(T) cycle, and the corresponding modes are LXTAL mode and PCLK mode.

The sampling time is configured through SPT bits in LPDTS_CFG register (see ***Table 40-2. Sampling time configuration***).

**Table 40-2. Sampling time configuration**

| SPT[3:0] | LXTAL or FM(T) clock cycle(s) |
|---|---|
| 0000 | 1 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |

| SPT[3:0] | LXTAL or FM(T) clock cycle(s) |
|---|---|
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | 10 |
| 1011 | 11 |
| 1100 | 12 |
| 1101 | 13 |
| 1110 | 14 |
| 1111 | 15 |

### 40.4.5. Trigger input

The trigger source can be selected to trigger the temperature measurement through ITSEL[3:0] bits in LPDTS_CFG.

- Software trigger

  The software trigger is selected when ITSEL[3:0] is set to '0000' in LPDTS_CFG.

  - Check if TSRF is set to 1. When TSRF is set, start the temperature measurement by setting TRGS bit in LPDTS_CFG register. otherwise, ignore this step.
  - After completion of measurement, if the TRGS bit remains at 1, the measurement will restart when the TSRF flag changes to 1.

- Hardware Trigger

  The temperature sensor can only capture a hardware trigger rising edge when TSRF bit is set, otherwise the trigger is ignored.

**Table 40-3. Trigger configuration**

| Name | ITSEL[3:0] | | | | Comment |
|---|---|---|---|---|---|
| NA | 0 | 0 | 0 | 0 | No hardware trigger |
| 0001 | 0 | 0 | 0 | 1 | reserved |
| 0010 | 0 | 0 | 1 | 0 | |
| 0011 | 0 | 0 | 1 | 1 | |
| 0100 | 0 | 1 | 0 | 0 | LPDTS_TRG |
| 0101 | 0 | 1 | 0 | 1 | reserved |
| 0110 | 0 | 1 | 1 | 0 | |
| 0111 | 0 | 1 | 1 | 1 | |
| 1000 | 1 | 0 | 0 | 0 | |
| 1001 | 1 | 0 | 0 | 1 | |
| 1010 | 1 | 0 | 1 | 0 | |
| 1011 | 1 | 0 | 1 | 1 | |
| 1100 | 1 | 1 | 0 | 0 | |

| Name | ITSEL[3:0] | | | | Comment |
|---|---|---|---|---|---|
| 1101 | 1 | 1 | 0 | 1 | |
| 1110 | 1 | 1 | 1 | 0 | |
| 1111 | 1 | 1 | 1 | 1 | |

**Note:** The LPDTS_TRG is the output of the TRIGSEL module. The INSELx[7:0] of TRIGSEL_LPDTS register in the TRIGSEL module are used to select trigger input souce of LPDTS_TRG trigger input.

### 40.4.6. On-off control and ready flag

The LPDTS block can be enabled by setting TSEN bit in LPDTS_CFG register. The TSRF flag in the Temperature sensor status register (LPDTS_STAT) indicate that the LPDTS block is ready for temperature measurement: when TSRF bit is set to 1, the measurement can be started. Once a measurement has started, TSRF bit is reset. New measurement can not be initiated at this time. If a new measurement is needed,it is necessary to wait for the completion of the last measurement and set the TSRF bit.

### 40.4.7. LPDTS low-power modes

**Table 40-4. Temperature sensor behavior in low-power modes**

| Mode | Description |
|---|---|
| Sleep | LPDTS interrupt can be used to exit from Sleep mode when the LXTAL or PCLK as reference clock. |
| Deep-sleep | LPDTS interrupt can be used to exit from Deep-sleep mode when the LXTAL as reference clock. |

### 40.4.8. LPDTS interrupts

The LPDTS interrupt line can be connected to the CPU NVIC or to the EXTI controller.

- The interrupt can be generated in two situations:
  - At the end of measurement.
  - The measurement result is higher or lower than a specified threshold.
- There are two kinds of interrupt in LPDTS module.
  - Synchronous interrupt: Three interrupt events can be select via 3 bits in LPDTS_INTEN register.
  - Asynchronous wakeup: Three asynchronous wakeup events can be selected via 3 bits in LPDTS_INTEN register.
- All combination of interrupts are allowed.

**Note:** Asynchronous wakeup is used only when the LXTAL is selected as reference clock.
The following table shows the interrupt bits and their description.

**Table 40-5. Temperature sensor behavior in low-power modes**

| Interrupt event | Interrupt flag | Enable control bit | Interrupt clear bit | Exit from Sleep mode | Exit from Deep-sleep mode | Synchronous/ Asynchronous |
|---|---|---|---|---|---|---|
| When the measurement is done | EMIF in LPDTS_STAT | EMIE in LPDTS_INTEN | EMIC in LPDTS_INTC | YES | NO | Synchronous on PCLK |
| When the measurement is lower than the specified threshold | LTIF in LPDTS_STAT | LTIE in LPDTS_INTEN | LTIC in LPDTS_INTC | YES | NO | |
| When the measurement is higher than the specified threshold | HTIF in LPDTS_STAT | HTIE in LPDTS_INTEN | HTIC in LPDTS_INTC | YES | NO | |
| When the measurement is done | EMAIF in LPDTS_STAT | EMAIE in LPDTS_INTEN | EMAIC in LPDTS_INTC | YES | YES | Asynchronous |
| When the measurement is lower than the specified threshold | LTAIF in LPDTS_STAT | LTAIFE in LPDTS_INTEN | LTAIC in LPDTS_INTC | YES | YES | |
| When the measurement is higher than the specified threshold | HTAIF in LPDTS_STAT | HTAIE in LPDTS_INTEN | HTAIC in LPDTS_INTC | YES | YES | |

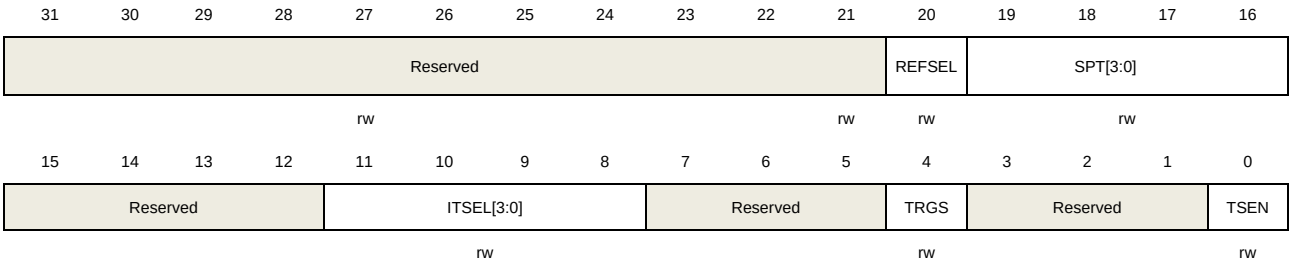
## 40.5. Register definition

LPDTS base address: 0x5800 6800

### 40.5.1. Configuration register (LPDTS_CFG)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



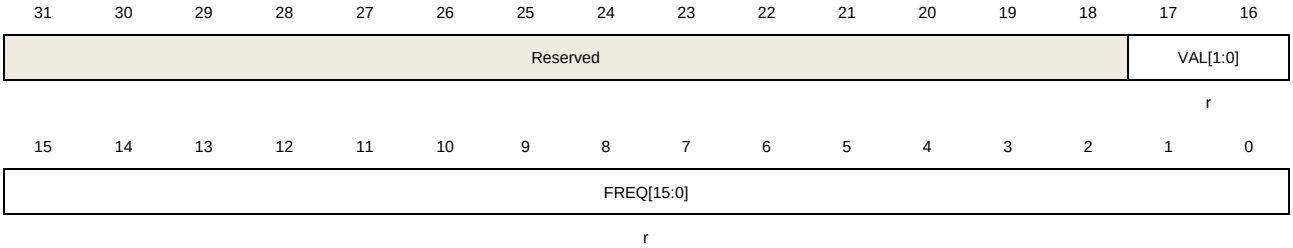

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | REFSEL | Reference clock selection<br>0: High speed reference clock (PCLK)<br>1: Low speed reference clock (LXTAL) |
| 19:16 | SPT[3:0] | Sampling time<br>These bits allow increasing the sampling time to improve measurement precision. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:8 | ITSEL[3:0] | Input trigger selection<br>These bits select which input triggers a temperature measurement. |
| 7:5 | Reserved | Must be kept at reset value. |
| 4 | TRGS | Trigger selection for frequency measurement<br>0: No software trigger.<br>1: Software trigger for a frequency measurement when temperature sensor is ready. |
| 3:1 | Reserved | Must be kept at reset value. |
| 0 | TSEN | Enable temperature sensor<br>0: Disable Temperature sensor<br>1: Enable Temperature sensor |

### 40.5.2. Sensor T0 data register (LPDTS_SDATA)

Address offset: 0x08

System reset value: 0x000X XXXX

This register has to be accessed by word (32-bit).



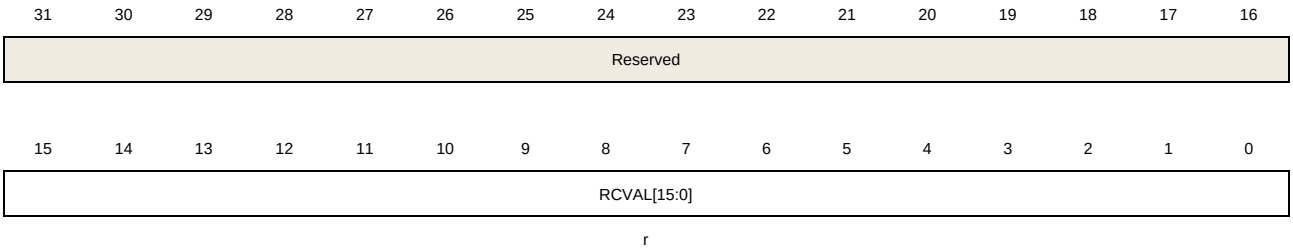

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17:16 | VAL[1:0] | Engineering value<br>These bits are the value of T0 temperature.<br>0x00: 25 °C<br>0x01: -40 °C<br>Others: Reserved |
| 15:0 | FREQ[15:0] | Frequency value<br>These bits are the value of the frequency measured when the temperature is T0.<br>This value is set by 0.1 kHz step. |

### 40.5.3. Ramp data register (LPDTS_RDATA)

Address offset: 0x10

System reset value: 0xXXXX XXXX

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | RCVAL[15:0] | Ramp coefficient<br>These bits are the value of the ramp coefficient. |

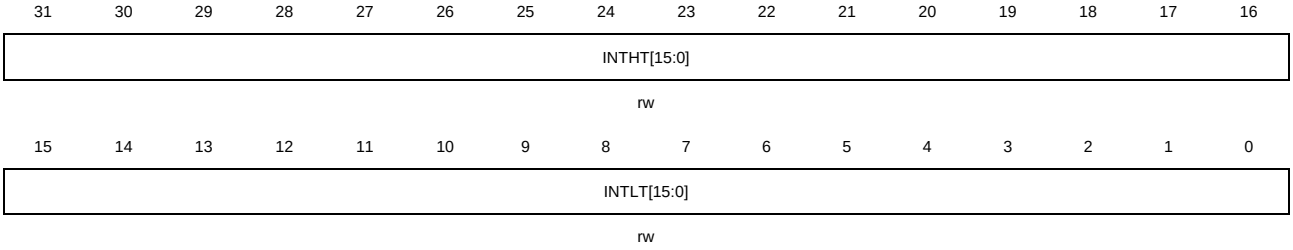
This value is set by 1 Hz/°C step.

### 40.5.4. Interrupt threshold register (LPDTS_IT)

Address offset: 0x14

System reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | INTHT[15:0] | Interrupt high threshold<br>These bits indicate the highest value than can be reached before the interrupt is generated. |
| 15:0 | INTLT[15:0] | Interrupt low threshold<br>These bits indicate the lowest value than can be reached before the interrupt is generated. |

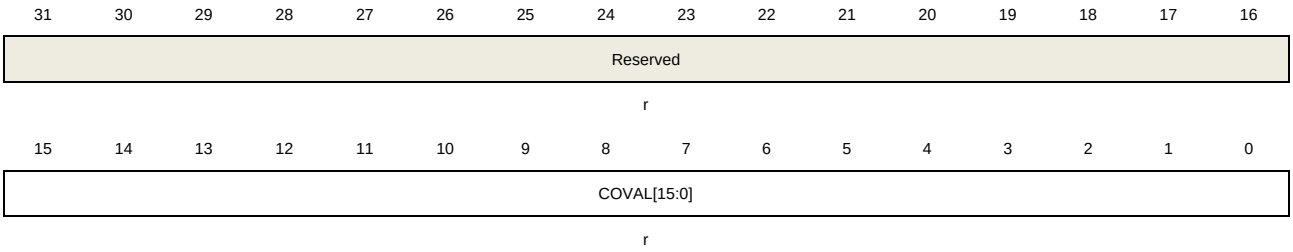
### 40.5.5. Temperature data register (LPDTS_DATA)

Address offset: 0x1C

System reset value: 0x0000 0000

This register contains the number of REF_CLK cycles used to compute the FM(T) frequency.

This register has to be accessed by word (32-bit).



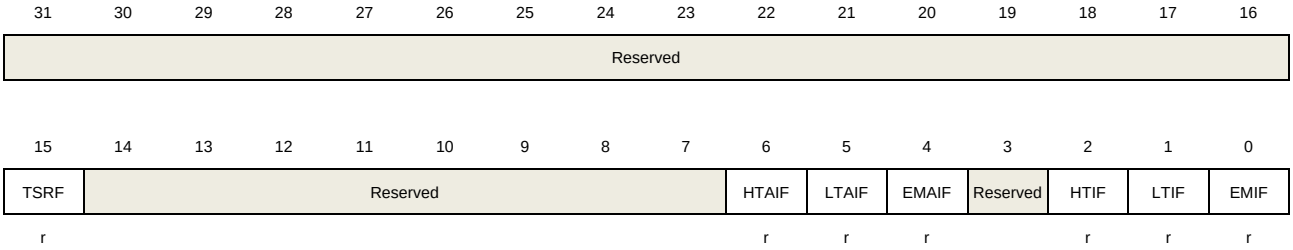

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | COVAL[15:0] | Value of the counter output |

### 40.5.6. Temperature sensor status register (LPDTS_STAT)

Address offset: 0x20

System reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | TSRF | Temperature sensor ready flag<br>0: Temperature sensor not ready<br>1: Temperature sensor ready |
| 14:7 | Reserved | Must be kept at reset value. |
| 6 | HTAIF | High threshold asynchronous interrupt flag<br>Set by hardware when the high threshold is reached and the HTAIE bit is set.<br>Reset by software by setting the HTAIC bit in the LPDTS_INTC register.<br>0: No high threshold asynchronous interrupt generated<br>1: High threshold interrupt asynchronous generated |
| 5 | LTAIF | Low threshold asynchronous interrupt flag<br>Set by hardware when the low threshold is reached and the LTAIE bit is set.<br>Reset by software by setting the LTAIC bit in the LPDTS_INTC register.<br>0: No low threshold asynchronous interrupt generated<br>1: Low threshold asynchronous interrupt generated |
| 4 | EMAIF | End of measurement asynchronous interrupt flag<br>Set by hardware when the temperature measure is finished and the EMAIE bit is set.<br>Reset by software by setting the EMAIC bit in the LPDTS_INTC register.<br>0: No end of measurement asynchronous interrupt generated<br>1: End of measurement asynchronous interrupt generated |
| 3 | Reserved | Must be kept at reset value. |
| 2 | HTIF | High threshold interrupt flag<br>Set by hardware when the high threshold is reached and the HTIE bit is set (synchronized on PCLK).<br>Reset by software by setting the HTIC bit in the LPDTS_INTC register.<br>0: No high threshold interrupt generated |

| | | |
|---|---|---|
| | | 1: High threshold interrupt generated |
| 1 | LTIF | Low threshold interrupt flag<br>Set by hardware when the low threshold is reached and the LTIE bit is set (synchronized on PCLK).<br>Reset by software by setting the LTIC bit in the LPDTS_INTC register.<br>0: No low threshold interrupt generated<br>1: Low threshold interrupt generated |
| 0 | EMIF | End of measurement interrupt flag<br>Set by hardware when the temperature measure is finished and the EMIE bit is set (synchronized on PCLK).<br>Reset by software by setting the EMIC bit in the LPDTS_INTC register.<br>0: No end of measurement interrupt generated<br>1: End of measurement interrupt generated |

### 40.5.7. Interrupt enable register (LPDTS_INTEN)

Address offset: 0x24

System reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | HTAIE | LTAIE | EMAIE | Reserved | HTIE | LTIE | EMIE |
| | | | | | | | | | rw | rw | rw | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | HTAIE | High threshold asynchronous interrupt enable<br>Set and reset by software to enable/disable the high threshold asynchronous interrupt (only when REFSEL = 1).<br>0: Disable the high threshold asynchronous interrupt<br>1: Enable the high threshold asynchronous interrupt |
| 5 | LTAIE | Low threshold asynchronous interrupt enable<br>Set and reset by software to enable/disable the low threshold asynchronous interrupt (only when REFSEL = 1).<br>0: Disable the low threshold asynchronous interrupt<br>1: Enable the low threshold asynchronous interrupt |
| 4 | EMAIE | End of measurement asynchronous interrupt enable<br>Set and reset by software to enable/disable the end of measurement asynchronous interrupt (only when REFSEL = 1). |

| | | |
|---|---|---|
| | | 0: Disable the end of measurement asynchronous interrupt<br>1: Enable the end of measurement asynchronous interrupt |
| 3 | Reserved | Must be kept at reset value. |
| 2 | HTIE | High threshold interrupt enable<br>Set and reset by software to enable/disable the high threshold interrupt which is synchronized on PCLK.<br>0: Disable the high threshold interrupt<br>1: Enable the high threshold interrupt |
| 1 | LTIE | Low threshold interrupt enable<br>Set and reset by software to enable/disable the low threshold interrupt which is synchronized on PCLK.<br>0: Disable the low threshold interrupt<br>1: Enable the low threshold interrupt |
| 0 | EMIE | End of measurement interrupt enable<br>Set and reset by software to enable/disable the end of measurement interrupt which is synchronized on PCLK.<br>0: Disable the end of measurement interrupt<br>1: Enable the end of measurement interrupt |

### 40.5.8. Interrupt clear flag register (LPDTS_INTC)

Address offset: 0x28

System reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | HTAIC | LTAIC | EMAIC | Reserved | HTIC | LTIC | EMIC |
| | | | | | | | | | rc_w1 | rc_w1 | rc_w1 | | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | HTAIC | High threshold asynchronous interrupt clear<br>Write 1 by software to clear the HTAIF flag in the LPDTS_STAT register. |
| 5 | LTAIC | Low threshold asynchronous interrupt clear<br>Write 1 by software to clear the LTAIF flag in the LPDTS_STAT register. |
| 4 | EMAIC | End of Measure asynchronous interrupt clear<br>Write 1 by software to clear the EMAIF flag of the LPDTS_STAT register. |

| | | |
|---|---|---|
| 3 | Reserved | Must be kept at reset value. |
| 2 | HTIC | High threshold interrupt clear<br>Write 1 by software to clear the HTIF flag in the LPDTS_STAT register. |
| 1 | LTIC | Low threshold interrupt clear<br>Write 1 by software to clear the LTIF flag in the LPDTS_STAT register. |
| 0 | EMIC | End of measurement interrupt clear<br>Write 1 by software to clear the EMIF flag in the LPDTS_STAT register. |

### 40.5.9. Option register (LPDTS_OP)

Address offset: 0x2C

System reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP [15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | OP [31:0] | general purpose option bits |

# 41. Encoder Divided-Output controller (EDOUT)

## 41.1. Overview

The encoder divided-output controller (EDOUT) is used to output location information obtained from the encoder in the form of A-phase, B-phase, and Z-phase pulses.

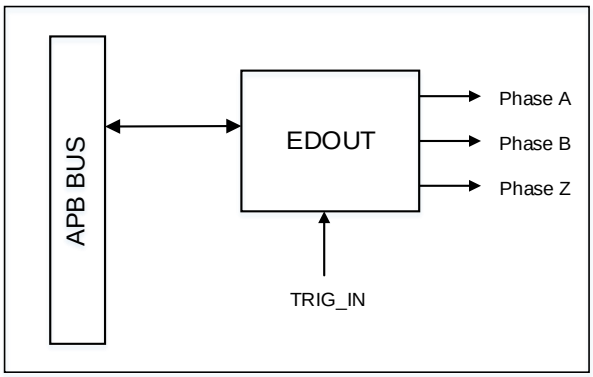
## 41.2. Characteristics

- Support for changing the activation polarity of B.
- Support configuration of Z-phase output location and pulse width.
- Number of edges per rotation: 16 to 65536 (must be the multiple of four).
- Support for the input of update period event signals from the TRIGSEL.

## 41.3. Function overview

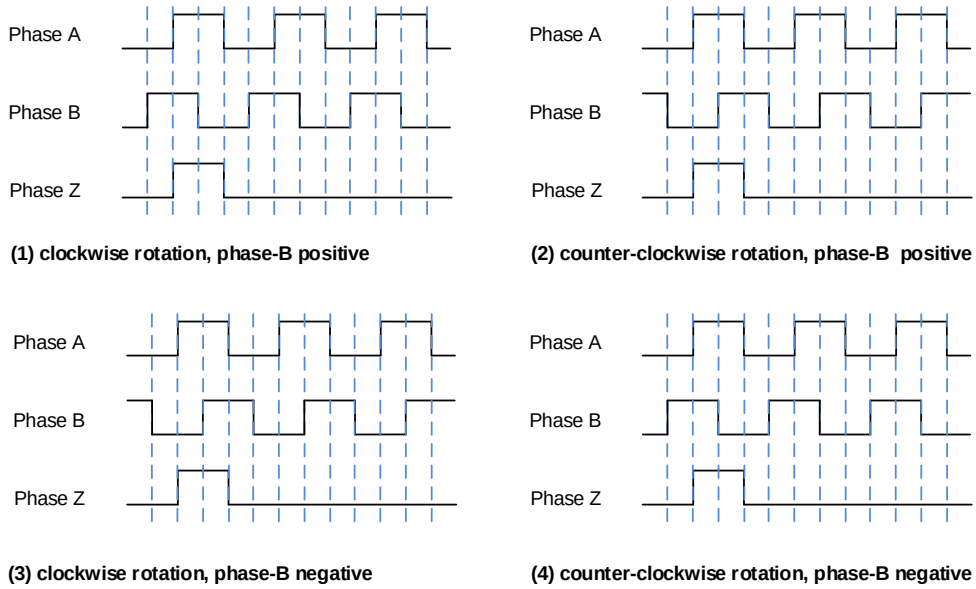
The EDOUT block diagram show as ***Figure 41-1. Block diagram of EDOUT***. In the figure, Phase A, Phase B and Phase Z represent the output pins of AB-phase a nd Z-phase respectively, and TRIG_IN represents the input signal from TRIGSEL.

**Figure 41-1. Block diagram of EDOUT**




The EDOUT output waveform is similar to the incremental encoder output signal. By setting registers, reflect the current location information. As shown in ***Figure 41-2. ABZ-phase output waveforms***.

**Figure 41-2. ABZ-phase output waveforms**




By system configuration, EDOUT converts location information from an incremental or absolute encoders to AB-phase and Z-phase output signals.

In addition to the initial setup of EDOUT, the CPU gets the location change from the encoder, updates the EDOUT register in each update period, and obtains the AB-phase and Z-phase output signals.

At this point, the event signal from the TIMER module must be input to EDOUT via TRIGSEL.

## 41.4. Z-phase output mode

Z-phase supports two operation modes:

- Operation mode 0

Z-phase output according to the current location. User sets the ZOSP and ZOWH bit fields in ZCR register. When current location (LOCCNT bit field in EDOUT_LCNT register reflect location information) match ZOSP, Z-phase starts to output signal and the pulse width is ZOWH edges.

- Operation mode 1

Z-phase output according to the number of edges. User sets the ZOSP and ZOWH bit fields in ZCR register. In each update period, when output edges match ZOSP, Z-phase starts to output signal and pulse width is ZOWH edges.

In the above two operation modes, if the ZOWH bit field is set to 0, it means that the Z-phase is not output. Note that the ZCR register must be configured during the EDOUT run (i.e. the EDOUTEN bit of EDOUT_ENABLE register is 1); otherwise, the Z-phase has no valid output.

## 41.5. Operation guidance

EDOUT outputs AB-phase and Z-phase signals based on register settings. The update period of the AB-phase and Z-phase output is determined by the input of the TRIGSEL. Generate update period using a timer, etc., and output a signal from TRIGSEL to EDOUT. The specific setting procedure are described below.

### 41.5.1. EDOUT initialization

The EDOUT initialization steps are as follows:

1. Configure the EDOUT relevant output pins
2. Initialization settings for the EDOUT

   Initialize EDOUT_CTL and EDOUT_LOC registers, for example, when the B-phase output is negative logic and the number of edges per rotation is 88 (4 × 22). Need to set the POL bit in EDOUT_CTL register to 1 and the LOCMAX bit field in EDOUT_LOC register to 87 (4*22-1).
3. Set the initial value in EDOUT_LCNT register

   The initial value is set to the initial location of the absolute or incremental encoder for conversion to the AB-phase and Z-phase signals represented by the value from 0 to LOCMAX in EDOUT_LCNT register. The formula for expressing location value with value from 0 to LOCMAX is [encoder location value]×[number of edges per rotation]/[resolution of encoder location value](rounded down).

   For example, when the number of edges in each rotation is 88, the resolution of the encoder location value is 20 bits (1048576), and the initial value is 931802. Need to set the LOCCNT in the EDOUT_LCNT register to 78 (931802 × 88 / 1048576).
4. Set the TRIGSEL

   For example, when setting TIMER to generate update period. Select the TIMER trigger output as EDOUT input. Certainly, EDOUT also support other triger source, but the high level width of trigger must greater than $T_{PCLK}$ of EDOUT.
5. Enable the AB-phase and Z-phase output.

   To enable EDOUT output, setting the EDOUTEN bit in the EDOUT_ENABLE register to 1.
6. Start the TIMER operation.

   According to the TRIGSEL setting in step 4, the TIMER starts counting and generates a update period signal.

### 41.5.2. EDOUT update processing

The EDOUT update processing steps are as follows:

1. Get location information.

   Get location information from absolute or incremental encoder.

2. Set the value in EDOUT_OCNT register

   Firstly, calculate the value to be set in the PDC and EDGC bit fields of EDOUT_OCNT register. The current location represented by the value from 0 to LOCMAX is calculated as follows:

   (m) = [location value obtained in step 1]×[number of edges per rotation]/[resolution of encoder location value] (rounded down)

   At this point, the value of EDGC bit field is:

   (n) = (m) – [the previous calculated value of (m)]

   When the absolute value of (n) is greater than half of the number of rotated edges, then:

   A. If (n) is positive, the value of EDGC is (n) – [number of edges per rotation].

   B. If (n) is negative, the value of EDGC is (n) + [number of edges per rotation].

   When EDGC bit field is not 0, PDC bit field is [update period]/[$T_{PCLK}$]/[absolute value of the EDGC bit field] (rounded down). When the EDGC bit field is 0, set PDC bit field is 0xFFFF.

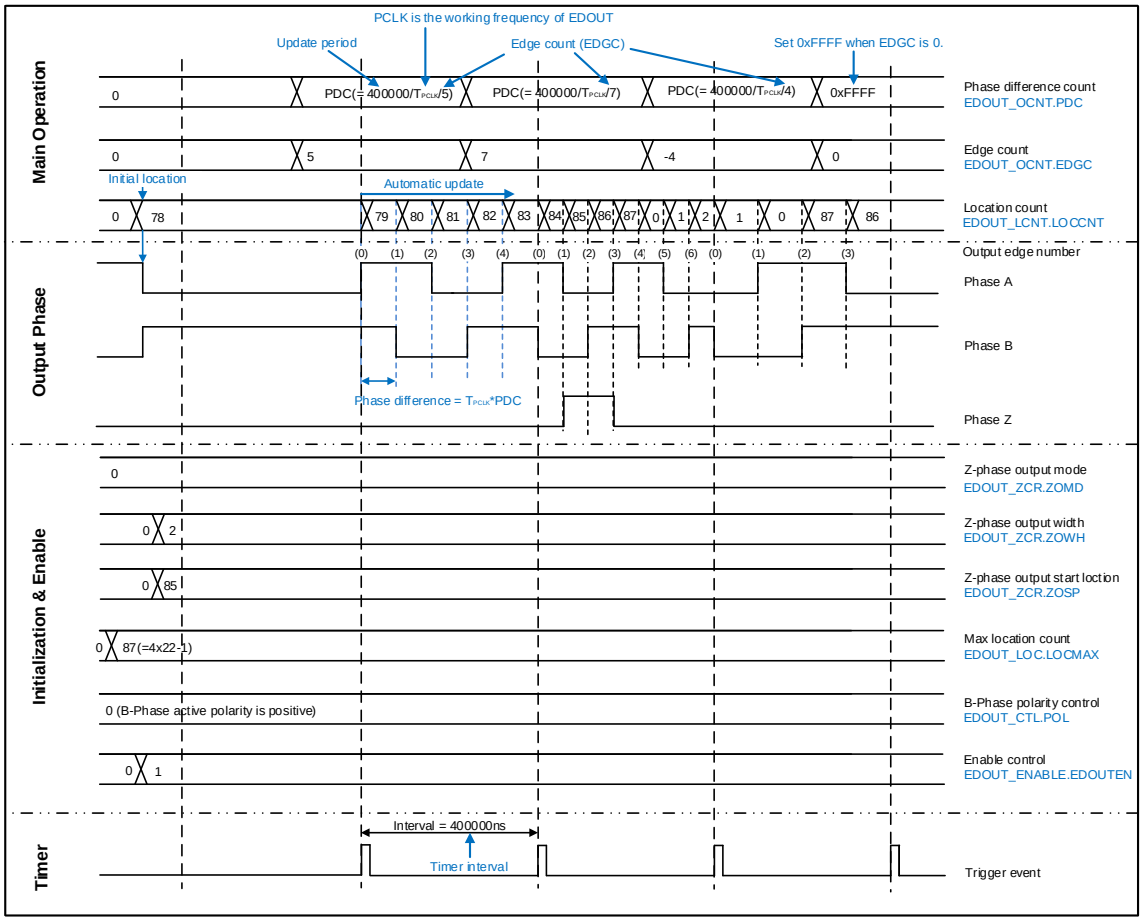
   Set the calculated PDC and EDGC bit fields value in the EDOUT_OCNT register.

### 41.5.3. EDOUT working example

As is shown in ***Figure 41-3. Example of the settings of EDOUT and the AB-Phase and Z-Phase output waveforms***. In this example, sets TIMER to generate update period and selects the TIMER trigger output as EDOUT input. The update period is 400000ns, the number of edges per rotation is 88, the B phase is positive, the Z phase output based on the current position, and the encoder location value represented from 0 to LOCMAX transition from the initial value of 78 to 83, 2 and 86.

**Figure 41-3. Example of the settings of EDOUT and the AB-Phase and Z-Phase output**

**waveforms**


PCLK is the working frequency of EDOUT
Update period
Edge count (EDGC)
Set 0xFFFF when EDGC is 0.
Main Operation
PDC(= 400000/TPCLK/5)
PDC(= 400000/TPCLK/7)
PDC(= 400000/TPCLK/4)
0xFFFF
Phase difference count
EDOUT_OCNT.PDC
Edge count
EDOUT_OCNT.EDGC
Initial location
Automatic update
Location count
EDOUT_LCNT.LOCCNT
Output edge number
Output Phase
Phase A
Phase B
Phase difference = TPCLK*PDC
Phase Z
Initialization & Enable
Z-phase output mode
EDOUT_ZCR.ZOMD
Z-phase output width
EDOUT_ZCR.ZOWH
Z-phase output start loction
EDOUT_ZCR.ZOSP
Max location count
EDOUT_LOC.LOCMAX
0 (B-Phase active polarity is positive)
B-Phase polarity control
EDOUT_CTL.POL
Enable control
EDOUT_ENABLE.EDOUTEN
Timer
Interval = 400000ns
Timer interval
Trigger event

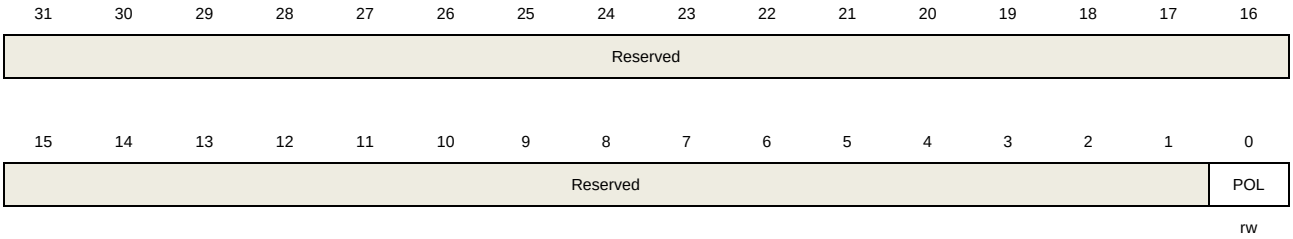
## 41.6. Register definition

EDOUT base address: 0x4001 8800

### 41.6.1. Control register (EDOUT_CTL)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

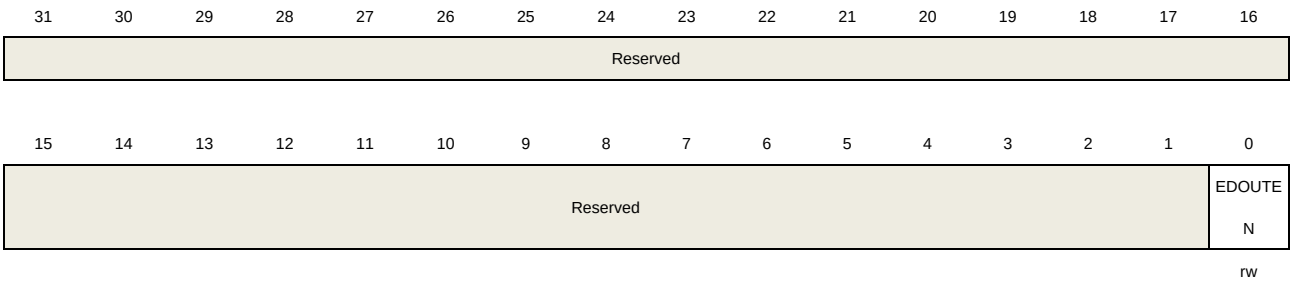



| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | POL | B-Phase active polarity<br>The active polarity of the B-phase output signal selection. If the EDOUTEN bit of EDOUT_ENABLE register is 0, the setting of this bit is reflected in the B phase output. Otherwise, this bit takes no effect.<br>0: Active polarity is positive<br>1: Active polarity is negative |

### 41.6.2. Enable register (EDOUT_ENABLE)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



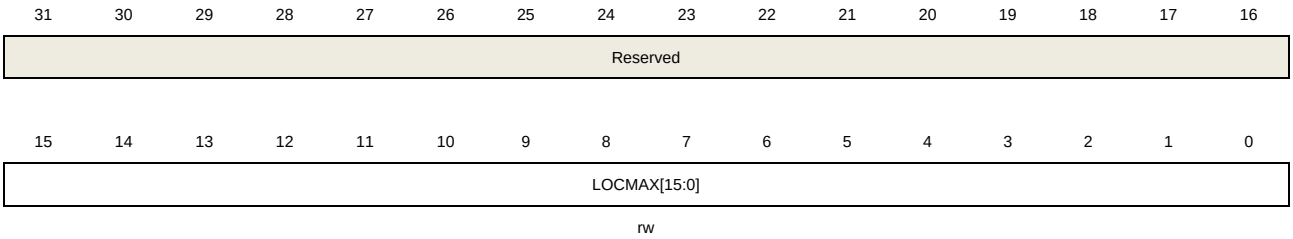

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | EDOUTEN | EDOUT enable bit. |

When this bit is set to 0, the AB-phase output the corresponding state immediately after the EDOUT_LCNT register is configured, the Z-phase output state is set to 0. When this bit is set to 1, the EDOUT starts and outputs the AB-phase and Z-phase signals.
0: Disabled EDOUT
1: Enabled EDOUT

### 41.6.3. Location register (EDOUT_LOC)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

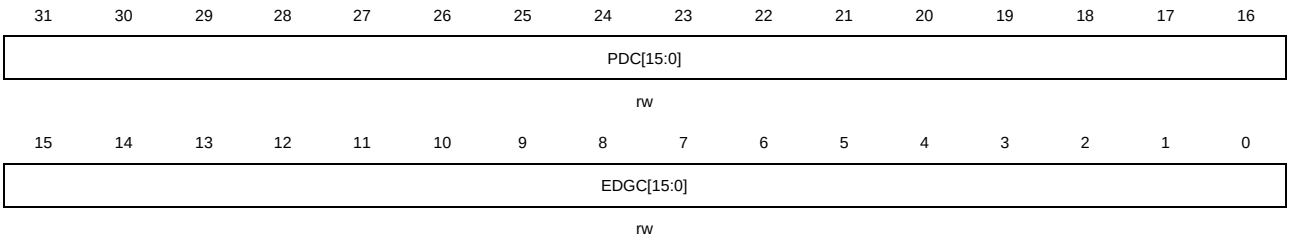



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | LOCMAX[15:0] | Maximum location value<br>These bits set the maximum location value for one rotation. The maximum location value is a multiple of four. If the maximum location value is “4×M”, set “4×M–1” in this register. The setting of these bits takes effect when the EDOUTEN bit of the EDOUT_ENABLE register is changed from 0 to 1.<br>0x0000~0x000E: Reserved<br>0x000F: The maximum location value is 16<br>...<br>0xyyyy: The maximum location value is 0xyyyy+1 |

### 41.6.4. Output counter register (EDOUT_OCNT)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | PDC[15:0] | Phase difference count<br>These bits set the phase difference between the A-phase signal and the B-phase signal for the next update period. The allowable range for setting value is 2 to 65535, in units of $T_{PCLK}$. When the EDGC bit field is set to 0, set these bits to 65535; when it is not 0, set them to "Update period / $T_{PCLK}$ / Absolute value of EDGC" (rounded down). When EDOUT is running (i.e. the EDOUTEN bit of EDOUT_ENABLE register is 1), please ensure that these bits are set before the next update period event signal is generated. |
| 15:0 | EDGC[15:0] | Edge count<br>These bits set the number of edges of the A-phase and the B-phase signal for the next update period. If you use reverse rotation, set a negative value that represents a two's complement. The allowable range for setting value are -32767(0x8001) to 32767(0x7FFF). The absolute value of these bits must not be greater than "Update period / (2*$T_{PCLK}$)". When EDOUT is running (i.e. the EDOUTEN bit of EDOUT_ENABLE register is 1), please ensure that these bits are set before the next update period event signal is generated. |

### 41.6.5. Location counter register (EDOUT_LCNT)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCCNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

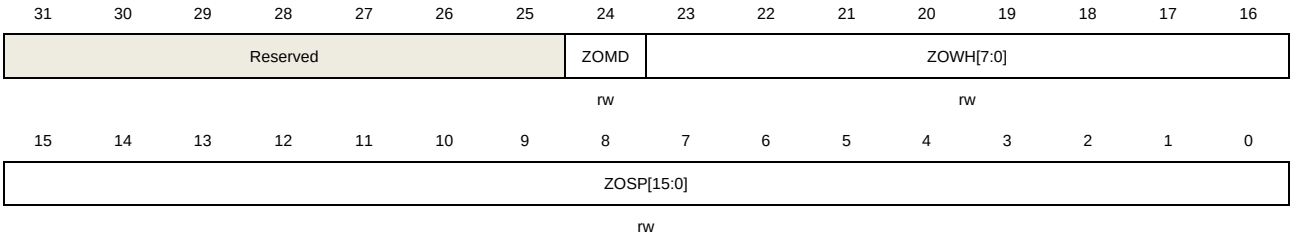
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | LOCCNT[15:0] | Current location value<br>These bits are used to set the current location value when EDOUT stops (i.e. the EDOUTEN bit of EDOUT_ENABLE register is 0). The allowable range for setting value is 0 to LOCMAX. After the current position is set, A-phase and B-phase will output the corresponding state immediately. When EDOUT is running (i.e. the EDOUTEN bit of EDOUT_ENABLE register is 1), these bits reflect the location value changes related to the A-phase and B-phase outputs. |

### 41.6.6. Z-phase configure register (EDOUT_ZCR)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | ZOMD | Z-phase output mode<br>0: Output according to the current location<br>1: Output according to the number of edges |
| 23:16 | ZOWH[7:0] | Z-phase output width |
| 15:0 | ZOSP[15:0] | Z-phase output start location |

# 42. Controller area network (CAN)

## 42.1. Overview

CAN bus (Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer. The CAN interface supports the CAN 2.0A/B protocol, ISO 11898-1:2015 and BOSCH CAN FD specification.

The CAN module is a CAN Protocol controller with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system consists of a set of mailboxes that store configuration and control data, timestamp, message ID, and data. The space of up to 32 mailboxes can also be configured as Rx FIFO with ID filtering against up to 104 extended IDs or 208 standard IDs or 416 partial 8-bit IDs, and configure receive FIFO/mailbox private filter register for up to 32 ID filter table elements.

**Note:** GD32H737xx series does not support CAN FD frames.

## 42.2. Characteristics

- Supports CAN protocol version 2.0A/B.
- Compliant with the ISO 11898-1:2015 standard.
- Supports CAN FD frames with up to 64 data bytes, baudrate up to 8 Mbit/s.
- Supports CAN classical frame with up to 8 data bytes, baudrate up to 1 Mbit/s.
- Supports time stamp based on 16-bit free running counter.
- Supports transmitter delay compensation for CAN FD frames at faster data rates.
- Maskable interrupts.
- Supports four communication mode: normal mode, Inactive mode, Loopback and silent mode, and Monitor mode.
- Supports two power saving modes: CAN_Disable mode, and Pretended Networking mode.
- Support two wakeup methods for waking up from Pretended Networking mode: wakeup matching event, and wakup timeout event.
- 32 mailboxes when configures with 8 bytes data length each, configurable as Rx or Tx mailbox.
- Global network time, synchronized by a specific message.
- The clock source of Protocol controller (CANCLK) is configurable(1).

**Transmission**

- Supports transmission abort.
- Tx mailbox status checkable.
- CRC for transmitted message.
- Supports priority of transmission message: lowest mailbox number, or highest priority.

**Reception**

- Receive private filter registers per Rx mailbox or Rx FIFO.
- Receive public filter register for Rx mailboxes and receive public filter register for Rx FIFO.
- Supports priority of message reception between mailboxes and Rx FIFO during matching process.
- Rx FIFO identifier filtering, supports identifier matching against either 104 extended, 208 standard, or 416 partial (8 bit) identifiers.
- Rx FIFO up to 6 frames depth, with DMA support.

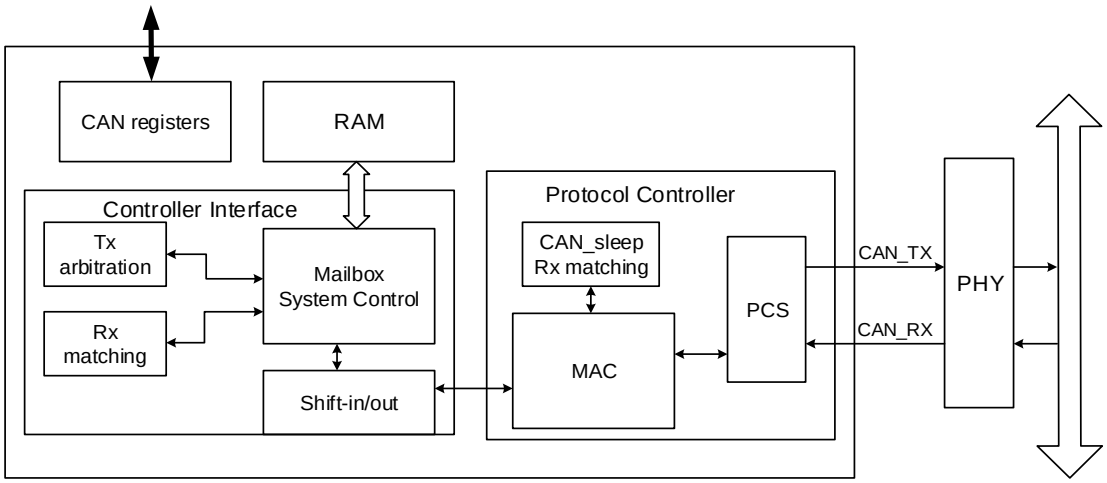
Note:

(1) The CANCLK source selection has restrictions. For details, please refer to the errata manual of the GD32H73x_75x series.

## 42.3. Function overview

*Figure 42-1. CAN module block diagram* shows the CAN block diagram.

**Figure 42-1. CAN module block diagram**




As shown in *Figure 42-1. CAN module block diagram*, CAN module includes three main parts:

- The Protocol controller

  The Protocol controller manages the communication on the CAN bus, including:

  MAC (Media Access Control):
  - Bit-stuffing/de-stuffing.
  - Stuff bit count for FD Frames.
  - Add CRC.
  - Construction of MAC frame.
  - ACK check/transmission.

  PCS (Physical Coding Sub-layer):
  - Bit timing.
  - Synchronization.

- TDC (Transmitter delay compensation).

Pretended Networking Rx matching:

- Process reception matching in Pretended Networking mode.

■ The Controller Interface

The Controller Interface manages RAM space selection for reception and transmission, including:

Tx arbitration:

- Find out the frame with the highest priority.

Rx matching:

- Compare the frame data received in the Rx shift buffer (an internal mailbox descriptor) with the fields in Rx mailbox or Rx FIFO according to the configured matching order.

Mailbox System Controller:

- Manage RAM space selection for reception and transmission, control the mailbox CODE, control the Rx FIFO pointer, and control the access requirement from the APB bus to the RAM space.

  The messages are stored in an embedded RAM dedicated to the CAN module. The dedicated RAM base address is module base address.

Shift in/out:

- Tranmit data between the selected mailbox / Rx FIFO descriptor and the Tx or Rx shift buffer.

■ CAN registers

The CAN registers is responsible for the CAN module communication with the APB bus.

### 42.3.1. Mailbox descriptor

The mailbox descriptor shown in ***Table 42-1. Mailbox descriptor with 64 byte payload*** can be used for both extended (29-bit identifier) and standard (11-bit identifier) frames. Each mailbox is formed by 16, 24, 40, or 72 bytes, depending on the data bytes allocation for the message payload: 8, 16, 32, or 64 data bytes, respectively. The memory area from offset 0x80 to 0x27F is used by the mailboxes.

**Table 42-1. Mailbox descriptor with 64 byte payload**

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MDES0 | FDF | BRS | ESI | Reserved | CODE[3:0] | | | | Reserved | SRR | IDE | RTR | DLC[3:0] | | | | TIMESTAMP[15:0] | | | | | | | | | | | | | | | |
| MDES1 | PRIO[2:0] | | | ID_STD[10:0] | | | | | | | | | | | | | ID_EXD[17:0] | | | | | | | | | | | | | | | |
| MDES2 | DATA_0[7:0] | | | | | | | | DATA_1[7:0] | | | | | | | | DATA_2[7:0] | | | | | | | | DATA_3[7:0] | | | | | | | |
| ... | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | | ... | | | | | | | |
| MDES17 | DATA_60[7:0] | | | | | | | | DATA_61[7:0] | | | | | | | | DATA_62[7:0] | | | | | | | | DATA_63[7:0] | | | | | | | |

**MDES0: Mailbox descriptor word 0**

Address offset: 0x80

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FDF | BRS | ESI | Reserved | CODE[3:0] | | | | Reserved | SRR | IDE | RTR | DLC[3:0] | | | |
| rw | rw | rw | | rw | | | | | rw | rw | rw | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMESTAMP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | FDF | FD format indicator<br>This bit is used to distinguish between CAN and CAN FD format frames.<br>For reception (Rx mailbox), no need to write this bit, it will be stored with the value received on the CAN bus. |
| 30 | BRS | Bit rate switch<br>This bit defines whether the bit rate is switched for a CAN FD frame.<br>For reception (Rx mailbox), no need to write this bit, it will be stored with the value received on the CAN bus. |
| 29 | ESI | Error state indicator<br>This bit indicates if the transmitting node is error active or error passive. This bit does not exist in Classical frames.<br>For transmission (Tx mailbox), it is transmitted dominant by error active nodes, and recessive by error passive nodes.<br>For reception (Rx mailbox), no need to write this bit, it will be stored with the value received on the CAN bus. |
| 28 | Reserved | Must be kept at rest value. |
| 27:24 | CODE[3:0] | Mailbox Code (CODE)<br>This bit field can be accessed by the CPU and by the CAN module, as part of the mailbox matching and arbitration process. The encoding is shown in ***Table 42-3. Mailbox Rx CODE*** and ***Table 42-4. Mailbox Tx CODE***. |
| 23 | Reserved | Must be kept at rest value. |
| 22 | SRR | Substitute remote request<br>This bit is only used in extended format. For transmission (Tx mailbox), it must be set to ‘1’ (recessive), if the bus transmits this bit as ‘0’ (dominant), then it means an arbitration loss. For reception (Rx mailbox), it will be stored with the value received on the CAN bus.<br>0: Not valid for transmission in extended format frames.<br>1: Transmission in extended format frames. |
| 21 | IDE | ID extended bit<br>This bit specifies whether the frame is standard or extended format.<br>For reception (Rx mailbox), it will be stored with the value received on the CAN bus.<br>0: Frame format is standard. |

1: Frame format is extended.

20 RTR

Remote transmission request

For transmission (Tx mailbox), if this bit is set to '1' (recessive), and the bus transmits this bit as '0' (dominant), then it means an arbitration loss. If this bit is set to '0' (dominant), and the bus transmits this bit as '1' (recessive), it is treated as a bit error. If the value configured matches the value transmitted, it is considered a successful bit transmission.

For reception (Rx mailbox), it will be stored with the value received on the CAN bus.

0: In Tx mailbox, the current mailbox has a data frame to be transmitted. In Rx mailbox, it may be considered in matching process.

1: In Tx mailbox, it means the current mailbox has a remote request frame to be transmitted. In Rx mailbox, incoming remote request frames may be stored.

**Note:** When configured as CAN FD frames, the RTR bit must be negated. This bit must be considered in classical frames only.

19:16 DLC[3:0]

Data length code in bytes

This bit field is the length (in bytes) of the Rx or Tx payload.

For reception (Rx mailbox), no need to write this bit field, they are written by the CAN module with the DLC field of the received frame.

For transmission (Tx mailbox), this bit field is written by the CPU with value of the frame to be transmitted. When RTR is 1, the frame to be transmitted is a remote request frame and does not include the data field, regardless of the DLC field.

15:0 TIMESTAMP[15:0]

Free-Running counter timestamp

This bit field is a copy of the free running counter, captured for Tx and Rx frames at the time when the beginning of the ID field appears on the CAN bus.

**Table 42-2. Data bytes for DLC**

| DLC | Data size in bytes |
|---|---|
| i (0 ≤ i ≤ 8) | i (0 ≤ i ≤ 8) |
| 9 | 12 |
| 10 | 16 |
| 11 | 20 |
| 12 | 24 |
| 13 | 32 |
| 14 | 48 |
| 15 | 64 |

**Table 42-3. Mailbox Rx CODE**

| CODE | Meaning | CODE after reception | Serviced(1) | RRFRMS(2) | Description |
|---|---|---|---|---|---|
| 0b0000 | INACTIVE | - | - | - | Mailbox does not participate in the matching process. |
| 0b0100 | EMPTY | FULL | - | - | After a frame is received successfully, the CODE field |

| CODE | Meaning | CODE after reception | Serviced(1) | RRFRMS(2) | Description |
|---|---|---|---|---|---|
| | | | | | is automatically switches to FULL. |
| 0b0010 | FULL | FULL | Yes | - | It remains FULL. If a new frame is moved to the mailbox after the mailbox was serviced, the code still remains FULL. |
| | | OVERRUN | No | | If the mailbox is FULL and a new frame is moved in before the CPU completes services it, the CODE field is automatically switches to OVERRUN. |
| 0b0110 | OVERRUN | FULL | Yes | - | If the CODE is OVERRUN and a new frame is moved in after CPU has serviced the mailbox, the CODE is automatically switches to FULL. |
| | | OVERRUN | No | | If the CODE field already indicates OVERRUN, and another new frame must be moved, the mailbox will be overwritten again, and the code will remain OVERRUN. |
| 0b1010 | RANSWER(3) | TANSWER (0x1110) | - | 0 | A Remote Answer was configured to recognize a remote request frame reception. After reception, the mailbox is set to transmit a response frame when RRFRMS bit in CAN_CTL2 register is 0. The code is automatically changed to TANSWER. |
| | | - | | 1 | The CODE is not effect during matching and arbitration process. |
| CODE[0] = 1 | BUSY(4) | FULL | - | - | Indicates that the mailbox is being updated. |
| | | OVERRUN | | | |

1. Serviced: Mailbox was serviced by CPU read, and was unlocked by reading CAN_TIMER register or other mailbox.
2. Remote Request Frame Stored bit, refer to ***Control register 2 (CAN_CTL2)***.
3. A mailbox with CODE 0b1010 should not be aborted. CODE 0b1010 must be used in mailbox which configured as CAN classical format, having the FDF bit reset.
4. If CODE[0] bit is set, the corresponding mailbox will not participate in the matching process. Notice that for Tx mailboxes, the BUSY bit should be ignored when read, except when MST bit in the CAN_CTL0 register is set.

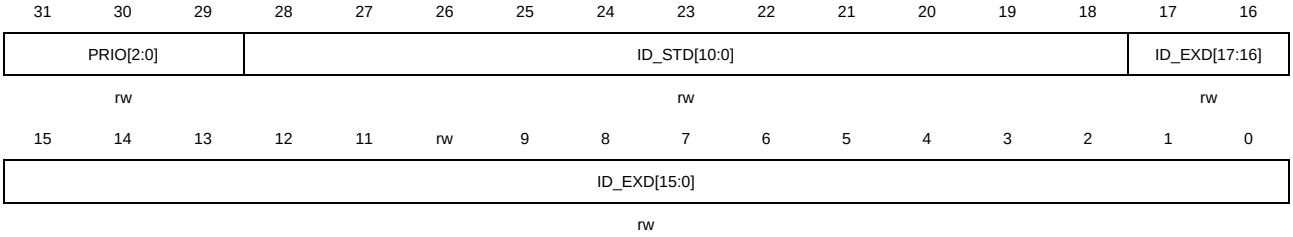
**Table 42-4. Mailbox Tx CODE**

| CODE | Meaning | CODE after transmission | RTR | Description |
|---|---|---|---|---|
| 0b1000 | INACTIVE | - | - | Mailbox does not participate in arbitration process. |
| 0b1001 | ABORT | - | - | Mailbox does not participate in arbitration process. |
| 0b1100 | DATA | INACTIVE | 0 | Transmit data frame. After transmission, the mailbox automatically returns to the INACTIVE state. |
| | REMOTE | EMPTY | 1 | Transmit remote request frame. After transmission, the mailbox automatically becomes an Rx empty mailbox |

| | | | | |
|---|---|---|---|---|
| | | | | with the same ID. |
| 0b1110 | TANSWER | RANSWER | - | This is an intermediate code which is automatically written to the mailbox by the controller interface when a matching remote request frame is received. After transmitting the remote response frame, the mailbox will automatically return to RANSWER state. The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit. |

### MDES1: Mailbox descriptor word 1

Address offset: 0x84




| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | PRIO[2:0] | Local priority<br>This bit field is only used when LAPRIOEN bit in CAN_CTL0 register is set.<br>This bit filed is only used for Tx mailboxes, while these bits are not transmitted, they are appended to the regular ID to define the transmission priority. |
| 28:18 | ID_STD[10:0] | Identifier for standard frame<br>In standard frame format, only these 11 most significant bits (28 to 18) are used for frame identification in both reception and transmission cases. The 18 least significant bits are ignored. |
| 17:0 | ID_EXD[17:0] | Identifier for extended frame<br>In extended frame format, ID_STD[10:0] & these bits are used for frame identification in both reception and transmission cases. |

### MDESx: Mailbox descriptor word x (x = 2..17)

Address offset: 0x80 + 0x04 * x (x = 2..17)




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:24 | DATA_i[7:0] | Data byte i (i = 4*x - 8)<br>Refer to DATA_i+3[7:0] descriptions. |
| 23:16 | DATA_i+1[7:0] | Data byte i+1 (i = 4*x - 8)<br>Refer to DATA_i+3[7:0] descriptions. |
| 15:8 | DATA_i+2[7:0] | Data byte i+2 (i = 4*x - 8)<br>Refer to DATA_i+3[7:0] descriptions. |
| 7:0 | DATA_i+3[7:0] | Data byte i+3 (i = 4*x - 8)<br>Up to 64 bytes can be used for a data frame, depending on the DLC value of the mailbox.<br>For Rx frames, the data received from the CAN bus are stored in this bit field. |

#### Mailbox number

When Rx FIFO is disabled, the dedicated RAM space is occupied by mailboxes only, so the mailbox number is the descriptor number which is incremented by one each time when across the complete mailbox descriptor length (with 8, 16, 32, or 64 data bytes).

When Rx FIFO is enabled (CAN FD mode disabled, data field is 8-byte length), the dedicated RAM space is occupied by both mailboxes and FIFO, so uniformly count the descriptor number by a mailbox descriptor length with 8 data bytes, then the mailbox number is the descriptor number which is occupied by mailbox.

#### Mailbox size for CAN FD

When CAN FD is enabled, the size of mailboxes that the CAN 512 bytes RAM can be partitioned is configured by MDSZ[1:0] bits in CAN_FDCTL register.

**Table 42-5. Mailbox size**

| MDSZ[1:0] | Payload size in bytes | Mailbox size |
|---|---|---|
| 0b00 | 8 | 32 |
| 0b01 | 16 | 21 |
| 0b10 | 32 | 12 |
| 0b11 | 64 | 7 |

### 42.3.2. Rx FIFO descriptor

The Rx FIFO descriptor is shown in ***Table 42-6. Rx FIFO descriptor***.

When RFEN bit in CAN_CTL0 register is 1, the RAM space which normally occupied by mailbox 0–5 with 8 byte payload is used for the Rx FIFO. FDES0 – FDES3 contains the output of the FIFO which is the oldest message that has been received but not yet read by the CPU. The RAM region 0x90-0xDC is reserved for internal use of the FIFO.

When RFEN bit in CAN_CTL0 register is 1, the RAM space which normally occupied by mailbox 6–31 with 8 byte payload is used for the ID filter table (configurable for 8 to up to 104

table elements) for receiving frames matching process into the FIFO. The ID filter table only contains 8 elements from FDES4 to FDES11 by default.

**Table 42-6. Rx FIFO descriptor**

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FDES0 | IDFMN[8:0] | | | | | | | | | SRR | IDE | RTR | DLC[3:0] | | | | TIMESTAMP[15:0] | | | | | | | | | | | | | | | |
| FDES1 | Reserved | | | | | | | | | ID_STD[10:0] | | | | | | | ID_EXD[17:0] | | | | | | | | | | | | | | | |
| FDES2 | DATA_0[7:0] | | | | | | | | | DATA_1[7:0] | | | | | | | DATA_2[7:0] | | | | | | | | | | | | | | | |
| FDES3 | DATA_4[7:0] | | | | | | | | | DATA_5[7:0] | | | | | | | DATA_6[7:0] | | | | | | | | | | | | | | | |
| 0x90 - 0xDC | Reserved | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDES4 | ID filter table element 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| … | … | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| FDES107 | ID filter table element 103 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

## FDES0: Rx FIFO descriptor word 0

Address offset: 0x80

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDFMN[8:0] | | | | | | | | | SRR | IDE | RTR | DLC[3:0] | | | |
| r | | | | | | | | | r | r | r | r | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMESTAMP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

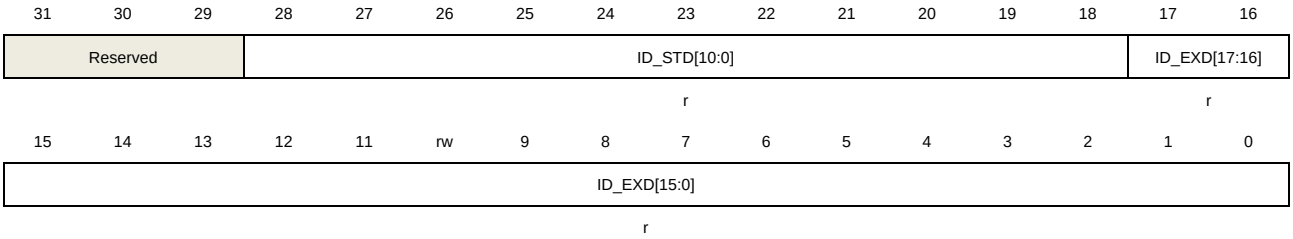
| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | IDFMN[8:0] | Identifier filter matching number<br>This bit field indicates which ID filter table element matches the received message that is in the output of the Rx FIFO. |
| 22 | SRR | Substitute remote request<br>This bit is only used in extended format. It will be stored with the value received on the CAN bus. |
| 21 | IDE | ID extended bit<br>This bit specifies whether the frame is standard or extended format.<br>0: Frame format is standard.<br>1: Frame format is extended. |
| 20 | RTR | Remote transmission request<br>0: Data frames are accepted<br>1: Remote frames are accepted |
| 19:16 | DLC[3:0] | Data length code in bytes |

This bit field is the length (in bytes) of the Rx payload.
For reception, this bit field is written by the CAN module with the DLC field of the received frame.

15:0 TIMESTAMP[15:0] Free-Running counter timestamp
This bit field is a copy of the free running counter, captured for Rx frames at the time when the beginning of the ID field appears on the CAN bus.

### FDES1: Rx FIFO descriptor word 1

Address offset: 0x84




| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at rest value. |
| 28:18 | ID_STD[10:0] | Identifier for standard frame<br>In standard frame format, only these 11 most significant bits (28 to 18) are used for frame identification. The 18 least significant bits are ignored. |
| 17:0 | ID_EXD[17:0] | Identifier for extended frame<br>In extended frame format, ID_STD[10:0] & these bits are used for frame identification. |

### FDES2: Rx FIFO descriptor word 2

Address offset: 0x88



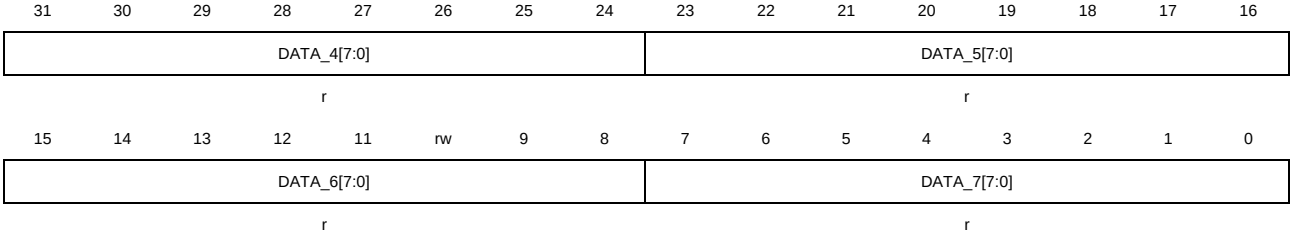

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DATA_0[7:0] | Data byte 0<br>Refer to DATA_3[7:0] descriptions. |
| 23:16 | DATA_1[7:0] | Data byte 1<br>Refer to DATA_3[7:0] descriptions. |
| 15:8 | DATA_2[7:0] | Data byte 2 |

Refer to DATA_3[7:0] descriptions.

| | | |
|---|---|---|
| 7:0 | DATA_3[7:0] | Data byte 3<br>Up to 8 bytes can be used for a data frame, depending on the DLC value of the mailbox. FD frames is not supported to receive in Rx FIFO. |

### FDES3: Rx FIFO descriptor word 3

Address offset: 0x8C




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DATA_4[7:0] | Data byte 4<br>Refer to DATA_7[7:0] descriptions. |
| 23:16 | DATA_5[7:0] | Data byte 5<br>Refer to DATA_7[7:0] descriptions. |
| 15:8 | DATA_6[7:0] | Data byte 6<br>Refer to DATA_7[7:0] descriptions. |
| 7:0 | DATA_7[7:0] | Data byte 7<br>Up to 8 bytes can be used for a data frame, depending on the DLC value of the mailbox. FD frames is not supported to receive in Rx FIFO. |

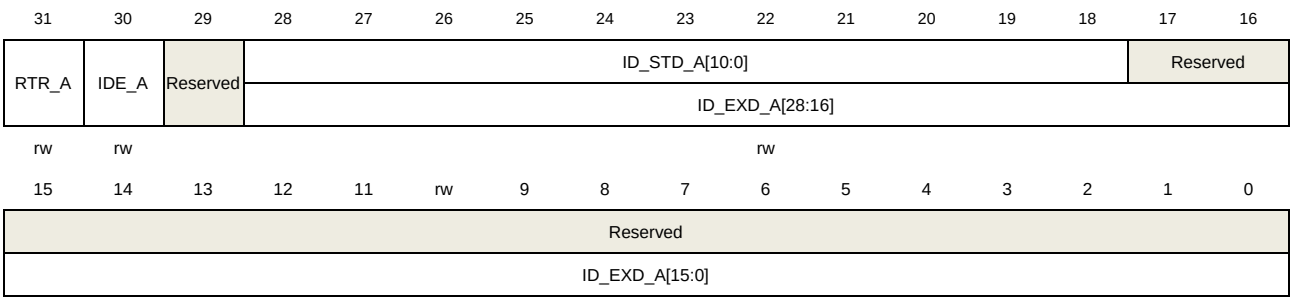
### FDESx: Rx FIFO descriptor word x (x = 4..107)

Address offset: 0xE0 + 4 * (x - 4)

This descriptor word shows the three different formats of the ID filter table elements, depending on the configuration of FS[1:0] bits in CAN_CTL0 register.

**Note:** The format is applied to all ID filter table elements. It is not possible to mix formats within the table.

**Format A mode:**



rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | RTR_A | Remote frame for format A<br>This bit specifies whether remote frames can be stored into the FIFO or not when matches.<br>0: It indicates that remote frames are rejected and data frames can be stored.<br>1: It indicates that remote frames can be stored and data frames are rejected. |
| 30 | IDE_A | ID Extended frame for format A<br>This bit specifies whether extended frames can be stored into the FIFO or not when matches.<br>0: Extended frames are rejected and standard frames can be stored.<br>1: Extended frames can be stored and standard frames are rejected. |
| 29 | Reserved | Must be kept at rest value. |
| 28:1 | ID_STD_A[10:0]/<br>ID_EXD_A[28:0] | ID in format A<br>This bit field specifies one full standard ID (standard or extended) for Rx FIFO matching process.<br>If IDE_A is 0, the 18 to 28 bits are used for standard ID, and the rest bits are reserved; otherwise, all these bits are used for extended ID. |

**Format B mode:**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTR_B0 | IDE_B0 | ID_STD_B_0[10:0] | | | | | | | | | | | Reserved | | |
| | | ID_EXD_B_0[13:0] | | | | | | | | | | | | | |
| rw | rw | rw | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | rw | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTR_B1 | IDE_B1 | ID_STD_B_1[10:0] | | | | | | | | | | | Reserved | | |
| | | ID_EXD_B_1[13:0] | | | | | | | | | | | | | |
| rw | rw | rw | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | RTR_B0 | Remote frame 0 for format B<br>This bit specifies whether remote frames can be stored into the FIFO or not when matches.<br>0: It indicates that remote frames are rejected and data frames can be stored.<br>1: It indicates that remote frames can be stored and data frames are rejected. |
| 30 | IDE_B0 | ID Extended frame 0 for format B<br>This bit specifies whether extended frames can be stored into the FIFO or not when matches.<br>0: Extended frames are rejected and standard frames can be stored.<br>1: Extended frames can be stored and standard frames are rejected. |

| | | |
|---|---|---|
| 29:16 | ID_STD_B_0[10:0]/ ID_EXD_B_0[13:0] | ID for frame 0 in format B<br>This bit field specifies a full standard ID or partial 14-bit extended ID for Rx FIFO matching process.<br>If IDE_B0 is 0, the 19 to 29 bits are used for standard ID, and the rest bits are reserved; otherwise, these bits are used for partial 14-bit extended ID, compared with the 14 most significant bits of the received ID. |
| 15 | RTR_B1 | Remote frame 1 for format B<br>Refer to RTR_B0 descriptions. |
| 14 | IDE_B1 | ID Extended frame 1 for format B<br>Refer to IDE_B0 descriptions. |
| 13:0 | ID_STD_B_1[10:0]/ ID_EXD_B_1[13:0] | ID for frame 1 in format B<br>Refer to ID_STD_B_0[10:0]/ ID_EXD_B_0[13:0] descriptions. |

**Format C mode:**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID_C_0[7:0] | | | | | | | | ID_C_1[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | rw | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ID_C_2[7:0] | | | | | | | | ID_C_3[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | ID_C_0[7:0] | ID for frame 0 in format C<br>This bit field specifies a partial 8-bit standard ID or partial 8-bit extended ID for Rx FIFO matching process.<br>In both standard and extended frame formats, the 8 bit field is compared with the 8 most significant bits of the received ID. |
| 23:16 | ID_C_1[7:0] | ID for frame 1 in format C<br>Refer to ID_C_0[7:0] descriptions. |
| 15:8 | ID_C_2[7:0] | ID for frame 2 in format C<br>Refer to ID_C_0[7:0] descriptions. |
| 7:0 | ID_C_3[7:0] | ID for frame 3 in format C<br>Refer to ID_C_0[7:0] descriptions. |

### 42.3.3. Communication modes

The CAN interface has four communication modes:

- Normal mode
- Inactive mode
- Loopback and silent mode

- Monitor mode

### Normal mode

In normal mode, the message reception and transmission, and errors are all managed normally, and all CAN protocol functions are enabled.

### Inactive mode

To enter Inactive mode, set INAMOD bit in CAN_CTL0 to 1 to enable Inactive mode, then set HALT bit in CAN_CTL0 register to 1 or put the chip into Debug mode.

When Inactive mode is requested, the following steps are performed before INAS bit asserted:

1. Wait for the bus 11 consecutive recessive bits.
2. Wait for the current transmission or reception processes being finished, it means all internal activities such as arbitration, matching, shift-in, and shift-out being finished. A pending shift-in does not prevent entering Inactive mode.
3. The Tx pin is driven as '1' (recessive).
4. Stop the prescaler.
5. Enable write access to the CAN_ERR0 register, which is read-only in other modes.
6. Set NRDY bit and INAS bit in CAN_CTL0 register.

When Inactive mode is entered, INAS bit in CAN_CTL0 register is set to 1 by CAN.

In Inactive mode, neither transmission nor reception is performed, and its prescaler is stopped, all registers are accessible.

To exit from Inactive mode, one of the following methods can meet:

- Clear INAMOD bit in CAN_CTL0.
- Clear HALT bit in CAN_CTL0 register, or the chip is removed from Debug mode.

If exiting from Inactive mode is requested, then INAS bit in CAN_CTL0 register is cleared after the CAN prescaler is running again. When out of Inactive mode, CAN module tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

**Note:** When in Inactive mode, the CAN_Disable mode request, or the Pretended Networking mode request will lead to INAS bit in CAN_CTL0 register be cleared and LPS bit in CAN_CTL0 register be set.

### Loopback and silent mode

To enter this mode, set the LSCMOD bit in CAN_CTL1 register to 1. In this mode, the messages are internally transmitted back to the receiver input, and the bit sent during the ACK slot in the frame acknowledge field is ignored to ensure reception transmitted by itself. Both transmit and receive interrupts are generated.

Loopback and silent mode is used for self-test. The Rx pin is ignored and the Tx pin holds in recessive state.

#### Monitor mode

To enter this mode, set MMOD bit in CAN_CTL1 register to 1.

When Monitor mode is entered, ERRSI[1:0] bit field in CAN_ERR1 register is set to 0b01 by CAN to indicate that the module works in an Error Passive state. In this mode, error counters are frozen for transmission and reception.

In Monitor mode, no transmission is performed and reception is performed only when messages are acknowledged by other CAN nodes, detection of a message that has not been acknowledged will lead to a bit dominant error flag (without changing the RECNT[7:0] or REFCNT[7:0] in CAN_ERR0 register).

### 42.3.4. Power saving modes

The CAN interface has two power saving modes:

- CAN_Disable mode.
- Pretended Networking mode.

In these two power saving modes, the dedicated RAM and the registers in SRAM can not be accessed.

#### CAN_Disable mode

The CAN module is enabled or disabled by configuring the CANDIS bit in CAN_CTL0 register.

For power saving, if CANDIS bit is set to 1 to disable CAN module, the CAN module will enter CAN_Disable mode after a delay when the LPS bit and NRDY bit in CAN_CTL0 register are changed to 1.

When CAN is disabled, the clocks to the Protocol controller and the Controller Interface are disabled. All registers except the CAN_RMPUBF, CAN_RFIFOPUBF, CAN_RFIFOIFMN, and CAN_RFIFOMPFx (x=0..31) registers are accessible. Also the dedicated RAM can not be accessed.

After CAN is enabled, you need to dalay a time to wait for LPS bit in CAN_CTL0 register to be cleared for Protocol controller recognization. When CAN is enabled, CAN module requests to resume the clocks to the Protocol controller and the Controller Interface.

#### Pretended Networking mode

Pretended Networking mode is used to receive wakeup messages with low power consumption. This mode can work together with MCU deepsleep mode.

To enter Pretended Networking mode, set PNEN bit and PNMOD bit in CAN_CTL0 register to 1, and optionally put the MCU into deepsleep mode.

When Pretended Networking mode is requested, the following steps are performed:

1. Wait for the bus to be in Idle state, or else wait for the third bit of Intermission, and then checks it to be recessive.
2. Set LPS bit and PNS bit in CAN_CTL0 register.
3. Request to disable the Controller Interface clock, while keeping the Protocol controller clock running.

In Pretended Networking mode, Controller Interface clock is disabled and Protocol controller is kept clocked (if the MCU works in deepsleep mode, the clock of CAN Protocol Controller should be configured as IRC64MDIV in advance, otherwise the the clock of CAN Protocol Controller will be lost), so that the reception process is still active to filter messages. The matching, arbitration, shift-in and shift-out processes are not performed in Pretended Networking mode.

To exit from Pretended Networking mode, the following method can meet:

- When a wakeup event is detected, and a wake up interrupt is occured. Clear PNMOD bit and PNEN bit in CAN_CTL0 register.
- Clear PNMOD bit and PNEN bit in CAN_CTL0 register.

If exiting from Pretended Networking mode is requested, CAN module will wait for the bus to be in Idle state or else wait for the third bit of Intermission to clear LPS bit and PNS bit in CAN_CTL0 register, and resume normal mode, CAN module will be synchronized to the CAN bus.

### 42.3.5. Data transmission

For transmission, an arbitration mechanism decides whether the Tx mailbox transmission priority is depending on the message ID (the PRIO field can also be configured to participate in arbitration), or on the mailbox number.

The quantity of mailboxes in CAN FD format is determined by MDSZ[1:0] bits in CAN_FDCTL register, refer to ***Table 42-5. Mailbox size***.

#### Transmit process

To transmit a CAN frame, a Tx mailbox must be prepared for transmission in following steps:

1. Check whether the corresponding mailbox state MSx bit in CAN_STAT register is set and clear it.
2. If the mailbox is active (either Tx or Rx), inactivate the mailbox by method described in ***Tx mailbox inactivation*** or ***Rx mailbox inactivation***, when Tx mailbox inactivation is performed, do the following steps, when Rx mailbox inactivation is performed, go to step 6. While if the mailbox is inactive (either Tx or Rx), go to step 6.
3. Poll the the corresponding MSx bit in CAN_STAT register to be set, or by the interrupt when MIEx bit in CAN_INTEN register is set.
4. Read back the CODE field to get the state of the mailbox (aborted, or transmitted).
5. Clear the corresponding flag MSx in the CAN_STAT register.
6. Write mailbox ID field (plus the mailbox PRIO field if LAPRIOEN bit in CAN_CTL0 register

is set to 1) of the MDES1 word.

7. Write payload data bytes in mailbox DATA field of MDESx (x = 2..17) word.
8. Configure the mailbox IDE, RTR, FDF, BRS, ESI, and DLC field to MDES0 word.
9. Activate the mailbox to transmit the frame by setting mailbox CODE field to 0b1100. When the mailbox is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the mailbox payload size is less than the mailbox DLC value, CAN adds the necessary number of bytes with constant 0xCC to meet the expected DLC.

Upon a successful transmission, the CODE field is automatically updated, and the TIMESTAMP field is automatically updated with the value of the free running counter; the CRC registers (CAN_CRCC and CAN_CRCCFD) are updated, and the corresponding flag MSx in the CAN_STAT register is set, if the interrupt enable bit MIEx in CAN_INTEN register is set, an interrupt will be generated.

### Arbitration process

When more than one Tx mailbox is pending, the arbitration process which searching from the lowest number mailbox to the higher ones will give the transmission order. The arbitration algorithm is controlled by the MTO bit in CAN_CTL1 register.

The arbitration process starts when matching one of the following situations:

- The CRC field on CAN bus: number of ASD[4:0] (in CAN_CTL2 register) CAN bits delay after the first bit of the CRC field.
- The Error or Overload Delimiter field on CAN bus.
- CAN bus is recovering from Bus Off state: number of ASD[4:0] (in CAN_CTL2 register) CAN bits delay after the counter TECNT[7:0] counted to 124. Recovering from Bus Off state needs 128 times of 11 continuous recessive bits, which is counted by TECNT[7:0] in CAN_ERR0 register.
- Exit from Inactive mode, or power saving mode (including CAN_Disable mode and Pretended Networking mode).
- Rewrite of MDES0 word of arbitration winner (temporary winner or final winner).
- Rewrite to MDES0 word of the scanned mailbox (arbitration is on-going): if no arbitration winner is found when scan finished, arbitration will restart at soon; otherwise, the arbitration process is finished.
- Write to MDES0 word of a mailbox: when no arbitration is processing, and no arbitration winner exists, and the CAN bus is not in SOF-DATA field / SOF-Control field of a data / remote frame or Error / Overload flag field of an Error / Overload frame.
- CAN node enters Bus Integration state (refer to ***Bus integration state***): Number of ASD[4:0] (in CAN_CTL2 register) CAN bits delay after the state.

Arbitration process stops when matching one of the following situations:

- All mailboxes are scanned.
- A Tx active mailbox is found when MTO bit in CAN_CTL1 register is set to 1 (lowest-number mailbox first).

- The Error or Overload flag field on CAN bus.
- The SOF field of the next frame on CAN bus.
- When Inactive mode, CAN_Disable mode or Pretended Networking mode is requested.

**Lowest-number mailbox first**

If MTO bit in CAN_CTL1 register is set to 1, the lowest number Tx mailbox is transmitted first, and LAPRIOEN bit in CAN_CTL0 register has no effect.

**Highest-priority mailbox first**

If MTO bit in CAN_CTL1 register is set to 0, then the Tx mailbox with the highest priority is transmitted first. The highest priority Tx mailbox is the one that has the lowest arbitration value (refer to ***Table 42-7. Mailbox arbitration value(32 bit) when local priority disabled*** and ***Table 42-8. Mailbox arbitration value(35 bit) when local priority enabled***) among all Tx mailboxes. If more than one mailboxes have equivalent arbitration values, the mailbox with the lowest number is the arbitration winner.

When LAPRIOEN bit in CAN_CTL0 register is set to 1, the local priority is disabled, the bits participate in the internal arbitration process are exactly what will be transmitted to the CAN bus, shown in ***Table 42-7. Mailbox arbitration value(32 bit) when local priority disabled***.

When LAPRIOEN bit in CAN_CTL0 register is set to 1, the local priority is enabled, then the mailbox PRIO field will participate in the internal arbitration process. Shown in ***Table 42-8. Mailbox arbitration value(35 bit) when local priority enabled***, the mailbox PRIO field is the most significant part of the arbitration value, thus mailboxes with low PRIO values have higher priority regardless of the rest of their arbitration values, while the PRIO field will not be transmitted to the CAN bus.

**Table 42-7. Mailbox arbitration value(32 bit) when local priority disabled**

| IDE | 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | | | | |
|---|---|---|---|---|---|
| 0 | ID_STD[10:0] | RTR | IDE | Reserved | |
| 1 | ID_EXD[28:18] | SRR | IDE | ID_EXD[17:0] | RTR |

**Table 42-8. Mailbox arbitration value(35 bit) when local priority enabled**

| IDE | 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 | | | | | |
|---|---|---|---|---|---|---|
| 0 | PRIO[2:0] | ID_STD[10:0] | RTR | IDE | Reserved | |
| 1 | PRIO[2:0] | ID_EXD[28:18] | SRR | IDE | ID_EXD[17:0] | RTR |

## Arbitration start delay

Arbitration start delay is configured by ASD[4:0] bits in CAN_CTL2 register to optimize the

transmission performance when the arbitration process ends too early to give a chance for the CPU to overwrite the winner Tx mailbox, thus the arbitration is restarted and may not be able to transmit in time.

### Shift-out

The shift-out process is the copy operation of the content from a Tx mailbox to the Tx shift buffer (an internal mailbox descriptor) after the arbitration winner is found. The message on the Tx shift buffer is transmitted according to the CAN protocol rules.

When the shift-out process is done, write access to the MDES0 word of the corresponding mailbox is blocked even if MST bit in CAN_CTL0 register is set. The write access to MDES0 word of the corresponding mailbox is recoverd when matching one of the following situations:

- After the mailbox is transmitted and the corresponding flag MSx in the CAN_STAT register is cleared by the CPU.
- CAN node enters Inactive mode or Bus Off state.
- CAN node loses the bus arbitration or there is an error during the transmission.

The shift-out process starts when matching one of the following situations:

- The first bit of Intermission field on CAN bus.
- During Bus Idle state.
- During Wait For Bus Idle state.

During the shift-out process, the CPU has priority to access the corresponding memory in Bus Idle state, and the shift-out operation has the lowest priority to access the corresponding memory.

### Abort

To request an abort of the transmission, the recommended operation is setting MST bit in the CAN_CTL0 register to 1, then writing ABORT (0b1001) to the CODE field of the mailbox.

The writing of ABORT (0b1001) to the mailbox MDES0 word is successfully when the mailbox is not the arbitration winner, or when the mailbox is the arbitration winner but the shift-out of the mailbox is not finished yet. In this situation, the corresponding MSx bit in CAN_STAT register will be set.

The writing of ABORT (0b1001) to the mailbox MDES0 word is blocked when shift-out of the mailbox is already finished, or when the mailbox is being transmitted. In this situation, the abort request is captured and kept pending until the frame is transmitted successfully or failed:

- The frame is transmitted successfully, the mailbox is not aborted: If the frame is transmitted successfully, the pending abort request will be cleared automatically, the corresponding MSx bit in CAN_STAT register will be set, and an interrupt will occur when MIEx bit in CAN_INTEN register is set.
- The frame is transmitted failed, the mailbox is aborted: If the frame failed to be transmitted, the pending abort request is responsed, the write access to the mailbox is

recoverd, with ABORT code written to the mailbox MDES0 word, the corresponding MSx bit in CAN_STAT register will be set, and an interrupt will occur when MIEx bit in CAN_INTEN register is set.

When matching one of the following situations, the frame failed to be transmitted:

- Lose the bus arbitration.
- There is an error during the transmission.
- Enter Bus Off state.
- There is an overload frame.

#### Tx mailbox inactivation

The way to inactivate a Tx mailbox:

- Write the CODE field of the Tx mailbox MDES0 with ABORT. This is the recommended way for inactivation, which will not cause the unknown transmission.
  This operation must be done when MST bit in the CAN_CTL0 register is 1.

### 42.3.6. Data reception

For Classical CAN frames, reception through FIFO and mailbox are both supported.

For CAN FD frames, reception is only supported through mailbox.

#### Mailbox reception

For mailbox reception, a received frame will be stored into the mailbox only when the frame ID matches the mailbox ID programmed in the ID field (or the mailbox ID group when the filter registers are applied).

To receive a CAN frame into a mailbox, a mailbox must be prepared for reception in following steps:

1. If the mailbox is active (either Tx or Rx), inactivate the mailbox by method described in ***Tx mailbox inactivation*** or ***Rx mailbox inactivation***, when Tx mailbox inactivation is performed, do the following steps, when Rx mailbox inactivation is performed, go to step 4. While if the mailbox is inactive (either Tx or Rx), go to step 4.
2. Poll the the corresponding MSx bit in CAN_STAT register to be set, or by the interrupt when MIEx bit in CAN_INTEN register is set.
3. Read back the CODE field to make sure that the mailbox is aborted, or transmitted.
4. Clear the corresponding flag MSx in the CAN_STAT register.
5. Write the mailbox ID field of MDES1 word, and write IDE, RTR field of MDES0 word if needed.
6. Write the EMPTY (0b0100) to the CODE field of MDES0 word to activate the mailbox.

Upon a successful reception, all bits of the mailbox descriptor (DATA, ID, TIMESTAMP, SRR, IDE, RTR, FDF, BRS, ESI, DLC, CODE) are stored with the received data field or automatically updated, and the corresponding flag MSx in the CAN_STAT register is set, if the interrupt enable bit MIEx in CAN_INTEN register is set, an interrupt will be generated.

The TIMESTAMP field is automatically updated with the value of the free running counter at the time of the second bit of frame's ID field.

To service (read) a Rx mailbox, the recommended steps are shown as below:

1. Poll the corresponding flag MSx in the CAN_STAT register to be set, or by the interrupt when MIEx bit in CAN_INTEN register is set.
2. Read the mailbox MDES0 word, and poll until the BUSY bit (in CODE field) is 0. When the BUSY bit is 0, the read operation of the mailbox will lock the mailbox, so that to prevent the mailbox being overwritten.
3. Read the contents of the mailbox.
4. Clear the corresponding flag MSx in the CAN_STAT register.
5. Read CAN_TIMER register to unlock the mailbox.

**Rx mailbox locking**

A locking mechanism is only applied for Rx mailbox: For CODE field with Rx FULL or Rx OVERRUN, a CPU read to the mailbox MDES0 word will lock the mailbox, thus the locking will prevent a new matching frame overwriting it.

The locking will be released when reading the CAN_TIMER register (global unlocking operation), or when MDES0 word of any other mailbox is read. When unlocked, a shift-in process will start with the pending message (the same in Inactive mode, while when LPS bit in CAN_CTL0 register is 1, the shift-in process will be delayed until LPS bit changes to 0).

If the mailbox is not unlocked in time, while a new matching frame is coming, then the new frame will overwrite the Rx shift buffer without a notification of a lost message, and no error is recorded.

**Note:** Mailbox inactivation (write CODE with Rx INACTIVE, or Tx ABORT) has higher priority than locking.

**Rx mailbox inactivation**

The way to inactivate a Rx mailbox:

- Write the CODE field of the Rx mailbox MDES0 with INACTIVE (Tx INACTIVE or Rx INACTIVE). But this operation may lead to a result that a matched message lost without notice.

**Note:** The Rx mailbox inactivation will automatically unlocks the mailbox. There is no write protection for Rx FIFO.

### Rx FIFO reception

The Rx FIFO is 6-message deep. When RFEN bit in CAN_CTL0 register is 1, Rx FIFO is enabled for reception. Rx FIFO can only be used for reception, and must not be enabled when CAN FD mode is enabled. The Rx FIFO descriptor refers to ***Rx FIFO descriptor***. There is a powerful filter system provided to filter a group of identifiers, reducing the interrupt servicing workload. The number of Rx FIFO filters is configured in RFFN[3:0] bits of CAN_CTL2 register,

up to 32 filters are configured in CAN_RFIFOMPFx (x = 0..31) registers (if RPFQEN bit in CAN_CTL0 register is 1), or CAN_RFIFOPUBF and CAN_RFIFOMPFx (x = 0..31) registers (if RPFQEN bit in CAN_CTL0 register is 0).

Rx FIFO has unread messages: Only when MS5_RFNE bit in CAN_STAT register is 1, the FDES0-FDES3 words are valid to be read to get the received message. When MS5_RFNE bit in CAN_STAT register is 1, it means there is at least one frame available to be read from Rx FIFO. If the interrupt enable bit MIEx in CAN_INTEN register is set, an interrupt will be generated; while if DMAEN bit in CAN_CTL0 register is set, the MS5_RFNE flag will generate the DMA request and no Rx FIFO interrupt is generated.

- To service (read) Rx FIFO by CPU, the recommended steps are shown as below:

  1. Poll the flag MS5_RFNE in the CAN_STAT register to be set, or by the interrupt when MIE5 bit in CAN_INTEN register is set.
  2. Read the Rx FIFO FDES0-FDES3 words, and if needed read CAN_RFIFOIFMN register.
  3. Clear the flag MS5_RFNE in the CAN_STAT register. If there are more than one messages in the Rx FIFO, the act of clearing the flag will update the Rx FIFO FDES0-FDES3 words with the next message, and CAN_RFIFOIFMN register will be updated at the same time, the flag MS5_RFNE remains set, and an interrupt occurs again if enabled, repeat step 2 and 3 to get the received messages.

- To service (read) Rx FIFO by DMA controller, the recommended operation are shown as below :

  1. Configure and enable the DMA controller for Rx FIFO reception.
  2. Service (read) Rx FIFO by CPU until the flag MS5_RFNE in the CAN_STAT register is cleared, to avoid an additional DMA request after DMA mode is enabled.
  3. Set DMAEN bit in CAN_CTL0 register to 1 to enable DMA mode.
  4. Wait for the DMA request. When the flag MS5_RFNE in the CAN_STAT register is set, a DMA request is generated.
  5. Upon receiving the DMA request, the DMA will read the Rx FIFO FDES0 to FDES3. FDES3 word must be read to clear the flag MS5_RFNE in the CAN_STAT register, if there are more than one messages in the Rx FIFO, the act of reading FDES3 word will update the Rx FIFO FDES0-FDES3 words with the next message, and CAN_RFIFOIFMN register (should be read before FDES3) will be updated at the same time, the flag MS5_RFNE remains set, and a DMA request is generated again. Steps 4 and 5 are repeated.

**DMA mode**

DMA mode is supported for Rx FIFO reception when RFEN bit and DMAEN bit in CAN_CTL0 register are both set. When the DMA mode is enabled, the CPU must not read the Rx FIFO.

When DMA mode is enabled, Rx FIFO FDES0 to FDES3 words will be read by DMA controller when there is unread message in Rx FIFO, to get the received message. In this mode, Rx FIFO warning flag MS6_RFW bit and Rx FIFO overflow flag MS7_RFO bit in CAN_STAT

register are reserved.

Before disabling DMA mode by clearing DMAEN bit in CAN_CTL0 register, a clear FIFO operation (when RFEN bit in CAN_CTL0 register is 1, set MS0 in the CAN_STAT register to 1 in Inactive mode) must be performed to clear the Rx FIFO contents. The act of clearing FIFO will clear MS5_RFNE bit in the CAN_STAT register, and cancel the DMA request.

**Clear FIFO**

when Rx FIFO is enabled (RFEN bit in CAN_CTL0 register is 1), set MS0 bit in the CAN_STAT register to 1 in Inactive mode will clear the Rx FIFO contents, while the Rx FIFO flags will not be cleared (except in DMA mode), thus before clearing FIFO operation, the Rx FIFO must be serviced until the flag MS5_RFNE in the CAN_STAT register is cleared.

**Flag**

**Rx FIFO not empty**

When MS5_RFNE bit in CAN_STAT register is 1, it means there is at least one frame available to be read from Rx FIFO.

**Rx FIFO warning**

When MS6_RFW bit in CAN_STAT register is 1, it means the number of unread messages within the Rx FIFO is increased to five from four due to the reception of a new one, the Rx FIFO is almost full.

**Rx FIFO overflow**

When MS7_RFO bit in CAN_STAT register is 1, it means there is an incoming message lost because the Rx FIFO is full.

## Matching process

The matching process is searching for a Rx mailbox or Rx FIFO (when enabled) with an ID matching with the frame ID on CAN bus, also, the IDE and RTR field will participate in the matching.

The matching process starts when completes the DLC field reception.

The matching of mailboxes is affected by the RPFQEN bit. If the RPFQEN bit is 0, the first mailbox to be matched is the match winner, regardless of whether it is free-to-receive or non-free-to-receive. If the RPFQEN bit is 1, the first free-to-receive mailbox matched is the winner or the last non-free-to-receive mailbox matched is the winner.

**Searching process**

- When the Rx FIFO is enabled, the RFO bit in CAN_CTL2 register gives the searching order.
  - If RFO bit is set to 1, matching process starts from Rx mailbox to Rx FIFO. The Rx mailbox is searched from the lowest number mailbox to the higher ones.

Firstly, searching for a matching mailbox that is available for receiving. If the RPFQEN bit is 0, the first mailbox that matches is the winner, regardless of whether it is free-to-receive or non-free-to-receive. If the RPFQEN bit is 1, the first free-to-receive mailbox matched is the match winner. Rx FIFO is no longer searched in any of the above cases.

Secondly, If the RPFQEN bit is 1 and no free-to-receive mailbox is matched, but a matching non-free-to-receive mailbox is found, then the Rx FIFO is also searched to determine the winner. when the Rx FIFO is matched and is not full, the Rx FIFO is the matching winner; otherwise, the matching winner is the last non-free-to-receive matched Rx mailbox(leading to mailbox CODE OVERRUN).

Thirdly, if no matched Rx mailbox is found (means no free-to-receive matched mailbox, nor non-free-to-receive matched mailbox), then matching is processed on Rx FIFO. In this case, if the Rx FIFO is matched but it is full, it will leads to Rx FIFO overflow, while if the Rx FIFO is not matched (no matter it is full or not), the message will not be received.

- If RFO bit is set to 0, matching process starts from Rx FIFO to Rx mailbox. If the Rx FIFO matches the searching conditions, and is not full, then the Rx FIFO is the matching winner, regardless of searching for mailboxes. If Rx FIFO does not match or it is full, then matching is processed on Rx mailbox. The matching of mailboxes is affected by the RPFQEN bit. If the RPFQEN bit is 0, the first mailbox to be matched is the match winner, regardless of whether it is free-to-receive or non-free-to-receive. If the RPFQEN bit is 1, the first free-to-receive mailbox matched is the winner. If no free-to-receive mailbox is found, the last non-free-to-receive mailbox matched is the winner.

■ When the Rx FIFO is disabled, the matching process only searches the Rx mailboxes. Refer to the mailbox matching description above.

A free-to-receive Rx mailbox can be:

■ For a data frame reception, or a remote frame reception when RRFRMS bit in CAN_CTL2 register is 1, it can be: A mailbox with CODE field EMPTY; A mailbox with CODE field FULL or OVERRUN, which has already been serviced (read) and unlocked.

■ For a remote frame reception when RRFRMS bit in CAN_CTL2 register is 0, it can be a mailbox with CODE field RANSWER.

**Searching conditions for matched Rx mailbox**

Searching conditions for matched Rx mailbox, refers to ***Table 42-9. Rx mailbox matching***:

■ When the frame in Rx shift buffer is a data frame (RTR field is 0), Rx mailbox with CODE EMPTY, FULL, and OVERRUN will be searched:

- When IDERTR_RMF bit in CAN_CTL2 register is 0, it means the IDE field will be compared and RTR field will not be compared (regardless of bit 30 and bit 31 in related filter register). The ID field will be compared, using bit 0 to bit 28 filter data configurations in related filter register.
- When IDERTR_RMF bit in CAN_CTL2 register is 1, it means all the IDE, RTR and ID fields will be compared, using bit 0 to bit 28, bit 30, bit 31 filter data configurations in related filter register.

- When the frame in Rx shift buffer is a remote frame (RTR field is 1):
  - If RRFRMS bit in CAN_CTL2 register is 0, it indicates that the Rx mailbox with CODE RANSWER will be searched, and the IDE, ID field will be compared, using bit 0 to bit 28, bit 30 filter data configurations in related filter register.
  - If RRFRMS bit in CAN_CTL2 register is 1, it indicates that the matching process is the same as a data frame, so Rx mailbox with CODE EMPTY, FULL, and OVERRUN will be searched:
    When IDERTR_RMF bit in CAN_CTL2 register is 0, it means the IDE field will be compared and RTR field will not be compared (regardless of bit 30 and bit 31 in related filter register). The ID field will be compared, using bit 0 to bit 28 filter data configurations in related filter register.
    When IDERTR_RMF bit in CAN_CTL2 register is 1, it means all the IDE, RTR and ID fields will be compared, using bit 0 to bit 28, bit 30, bit 31 filter data configurations in related filter register.

**Table 42-9. Rx mailbox matching**

| Received bit | Configuration bit | | Field in mailbox descriptor for matching | | | |
|---|---|---|---|---|---|---|
| RTR | IDERTR_RMF (in CAN_CTL2 register) | RRFRMS (in CAN_CTL2 register) | IDE | RTR | ID | CODE |
| 0 | 0 | - | Compared(1) | Never(2) | Filtered(3) | EMPTY / FULL / OVERRUN |
| | 1 | | Filtered | | | EMPTY / FULL / OVERRUN |
| 1 | - | 0 | Compared | Never | Compared | RANSWER |
| | 0 | 1 | Compared | Never | Filtered | EMPTY / FULL / OVERRUN |
| | 1 | | Filtered | | | EMPTY / FULL / OVERRUN |

1. Compared: This field in Rx mailbox descriptor is always compared with the received bit, regardless of the filter data configurations in related filter register.
2. Never: This field in Rx mailbox descriptor is not compared with the received bit, regardless of the filter data configurations in related filter register.
3. Filtered: This field in Rx mailbox descriptor is compared with the received bit, using the filter data configurations in related filter register.

### Searching conditions for matched Rx FIFO

Searching conditions for matched Rx FIFO, refers to ***Table 42-10. Rx FIFO matching***:

- If the FS[1:0] bits in CAN_CTL0 register is 0 or 1, it means A or B format of filter table is adopted, then all the IDE, RTR and ID fields will be compared, using bit 0 to bit 31 filter data configurations in related filter register.
- If the FS[1:0] bits in CAN_CTL0 register is 2, it means C format of filter table is adopted,

then the IDE, RTR will not be compared (no these fields in FIFO descriptor) and ID fields will be compared, using bit 0 to bit 31 filter data configurations in related filter register.

- If the FS[1:0] bits in CAN_CTL0 register is 3, it means D format of filter table is adopted, then all frames are rejected.

**Table 42-10. Rx FIFO matching**

| Configuration bit | Field in Rx FIFO descriptor for matching | | |
|---|---|---|---|
| FS[1:0] (in CAN_CTL0 register) | IDE | RTR | ID |
| 0 | Filtered | | |
| 1 | Filtered | | |
| 2 | Never | | Filtered |
| 3 | Not match(1) | | |

(1) Not match: All frames are rejected.

### Shift-in

The shift-in process is the copy operation of the content from a Rx shift buffer (an internal mailbox descriptor) to a Rx mailbox or Rx FIFO that matched it.

When there is a matching descriptor found in the FIFO or in the Rx mailboxes, a shift-in process will be pending. The pending shift-in process starts to transfer when meets all of the following conditions:

- There is a matching winner for the frame in the Rx shift buffer.
- The CAN bus is in:
  - The second bit of Intermission field.
  - The first bit of an Overload frame.
- The target mailbox is not locked.

When the target mailbox of a pending shift-in process is unlocked during Inactive mode, the pending shift-in process starts to transfer. While if the unlocking occurs when LPS bit in CAN_CTL0 register is 1, the pending shift-in process will still be delayed until LPS bit changes to 0.

When the shift-in process is on-going, the BUSY bit (CODE[0]) of the target mailbox is set to indicate that the mailbox is being updated.

The shift-in process can be cancelled for a Rx mailbox, but can not be cancelled for the Rx FIFO. The shift-in process will be cancelled when matching one of the following situations:

- The target mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished.
- The Rx shift buffer receives a message transmitted by itself while the self reception is disabled by setting SRDIS bit in CAN_CTL0 register.
- There is a CAN protocol error.

When the shift-in process is done, Rx mailbox descriptor or Rx FIFO descriptor (if Rx FIFO is enabled) will be updated with the received message, and CAN_RFIFOIFMN will be updated if shift-in to the Rx FIFO, CODE field of Rx mailbox descriptor will be updated if shift-in to the Rx mailbox.

#### Filter data configuration

**When Rx FIFO is disabled:**

- If RPFQEN bit in CAN_CTL0 register is 0, then CAN_RMPUBF is used for all mailboxes.
- If RPFQEN bit in CAN_CTL0 register is 1, then CAN_RFIFOMPFx (x = 0..31) is used for mailboxes individually.

**When Rx FIFO is enabled:**

- If RPFQEN bit in CAN_CTL0 register is 0, then CAN_RMPUBF is used for all mailboxes, CAN_RFIFOPUBF and CAN_RFIFOMPFx (x = 0..31) are used for all the Rx FIFO ID filter table elements, and the value of these registers must be all the same.
- If RPFQEN bit in CAN_CTL0 register is 1, then CAN_RFIFOMPFx (x = 0..31) is used for the Rx FIFO ID filter table elements defined by RFFN[3:0] bits in CAN_CTL2 register and the mailboxes individually (because the Rx FIFO descriptor and the Rx mailbox descriptors can not occupy the same RAM space at the same time), CAN_RFIFOPUBF is used for the Rx FIFO ID filter table elements of the rest.

#### Self reception

When SRDIS bit in CAN_CTL0 register is 1, self reception is disabled, thus the frames transmitted by itself will not be received even if there is a matched Rx mailbox or Rx FIFO is matched, and no flag or interrupt will be generated. When the SRDIS bit is 0, it is allowed to receive a matching frame sent by itself.

### 42.3.7. Data reception in Pretended Networking mode

When PNEN bit and PNMOD bit in CAN_CTL0 register are configured to 1,the Pretended Networking mode is enabled, then the CAN is able to process received messages in MCU sleep mode. A wakeup event will wake up the CAN module from the Pretended Networking mode.

There are four groups of registers used for matched message storage: CAN_PN_RWMxCS, CAN_PN_RWMxI, CAN_PN_RWMxD0 and CAN_PN_RWMxD1 registers, group x from 0 to 3. Therefore, four messages can be stored at most (when NMM[7:0] bits in CAN_PN_CTL0 register is larger than or equal to 4), and only the latest messages will be stored. The group x indicates the message arrival order. If NMM[7:0] is less than 4, only NMM[7:0] value of messages can be stored, at groups from 0 to NMM[7:0] minus 1.

When the data length of the frame to be stored is less than 8 bytes, the padding values which continued to the received DATA field to be written to the CAN_PN_RWMxD0 and CAN_PN_RWMxD1 registers (x = 0..3) are zeroes. No timestamp is stored for wakeup

matched frames.

**Note:** When in Pretended Networking mode, CAN FD format messages are ignored.

### Wakeup interrupt

There are two types of wakeup interrupt events, including wakeup match event, and wakeup timeout event. Each interrupt event has a dedicated flag bit in the CAN_PN_STAT register, and a dedicated enable bit in CAN_PN_CTL0 register. The relationship is described in the *Table 42-11. Interrupt events*.

An wakeup interrupt can be generated when any type of the wakeup interrupt event occurs and enabled.

**Wakeup timeout event**

When CAN reaches the timeout value, a wakeup timeout event will occur. The timeout is configured by WTO[15:0] bits in CAN_PN_TO register.

**Note:** Even if the timeout value is reached, CAN module will not stop the message filtering process until the CPU wakes up.

**Wakeup match event**

When CAN receives the matched wakeup frame/frames within the timeout, the wakeup match event will occur. MMCNT[7:0] bits in CAN_PN_STAT register reflects the number of matched messages from the time of entering Pretended Networking mode to the time the CPU wakes up.

**Note:** Even if CAN receives the matched wakeup frame/frames, the timeout counter will not stop counting until the CPU wakes up.

### Frame matching

The fields of frame participate in the wakeup matching process are IDE, RTR, ID, DLC, and DATA field.

- When FFT[1:0] bit field in CAN_PN_CTL0 register is configured to 0, it means a wakeup match event occurs when a frame is received with all fields except DATA field, DLC field (that is IDE, RTR, and ID field matched) matched.
- When FFT[1:0] bit field in CAN_PN_CTL0 register is configured to 1, it means a wakeup match event occurs when a frame is received with all fields (that is IDE, RTR, ID, DLC, and DATA field matched) matched.
- When FFT[1:0] bit field in CAN_PN_CTL0 register is configured to 2, it means a wakeup match event occurs when a specified number (configured by NMM[7:0] bits in CAN_PN_CTL0 register) of frames are received with all fields except DATA field, DLC field (that is IDE, RTR, and ID field matched) matched.
- When FFT[1:0] bit field in CAN_PN_CTL0 register is configured to 3, it means a wakeup match event occurs when a specified number (configured by NMM[7:0] bits in CAN_PN_CTL0 register) of frames are received with all fields (that is IDE, RTR, ID, DLC,

and DATA field matched) matched.

**IDE field matching**

The IDE field of a matched message is the same as the configured expected IDE field in CAN_PN_EID0 register, using filter data in CAN_PN_IFEID1 register.

**RTR field matching**

The RTR field of a matched message is the same as the configured expected RTR field in CAN_PN_EID0 register, using filter data in CAN_PN_IFEID1 register.

**ID field matching**

- When IDFT[1:0] bit field in CAN_PN_CTL0 register is configured to 0, it means the ID field of a matched message is the same as the configured expected ID field in CAN_PN_EID0 register, using filter data in CAN_PN_IFEID1 register.
- When IDFT[1:0] bit field in CAN_PN_CTL0 register is configured to 1, it means the ID field of a matched message is larger than or equal to the configured expected ID field in CAN_PN_EID0 register. CAN_PN_IFEID1 register is not used.
- When IDFT[1:0] bit field in CAN_PN_CTL0 register is configured to 2, it means the ID field of a matched message is smaller than or equal to the configured expected ID field in CAN_PN_EID0 register. CAN_PN_IFEID1 register is not used.
- When IDFT[1:0] bit field in CAN_PN_CTL0 register is configured to 3, it means the ID field of a matched message is larger than or equal to the configured expected ID field in CAN_PN_EID0 register, and is smaller than or equal to the configured expected ID field in CAN_PN_IFEID1 register.

**DLC field matching**

- The DLC field of a matched message is larger than or qual to the configured expected DLC low threshold DLCELT[3:0] in CAN_PN_EDLC register, and lower than or qual to the configured expected DLC high threshold DLCEHT[3:0] in CAN_PN_EDLC register.

**DATA field matching**

- When DATAFT[1:0] bit field in CAN_PN_CTL0 register is configured to 0, it means the DATA field of a matched message is the same as the configured expected DATA field in CAN_PN_EDLx (x = 0,1) registers, using filter data in CAN_PN_DF0EDH0 and CAN_PN_DF1EDH1 registers.
- When DATAFT[1:0] bit field in CAN_PN_CTL0 register is configured to 1, it means the DATA field of a matched message is larger than or equal to the configured expected DATA field in CAN_PN_EDLx (x = 0,1) registers. CAN_PN_DF0EDH0 and CAN_PN_DF1EDH1 registers are reserved.
- When DATAFT[1:0] bit field in CAN_PN_CTL0 register is configured to 2, it means the DATA field of a matched message is smaller than or equal to the configured expected DATA field in CAN_PN_EDLx (x = 0,1) registers. CAN_PN_DF0EDH0 and CAN_PN_DF1EDH1 registers are reserved.

- When DATAFT[1:0] bit field in CAN_PN_CTL0 register is configured to 3, it means the DATA field of a matched message is larger than or equal to the configured expected DATA field in CAN_PN_EDLx (x = 0,1) registers, and is smaller than or equal to the configured expected DATA field in CAN_PN_DF0EDH0 and CAN_PN_DF1EDH1 registers.

  **Note:** In this case, all the two 8 bytes of the expected data register should be configured, when the DLC of the received message (DLC field matched) is less than 8 bytes, then in DATA field matching, the data that matching with the expected data is the received DATA field plus the padding value zeros.

### 42.3.8. CAN FD operation

Both ISO CAN FD (ISO11898-1 specification) and non-ISO (Bosch CAN FD Specification V1.0) CAN FD protocols are supported, but they are incompatible with each other, so select the protocol by ISO bit in CAN_CTL2 register. In comparison to the non-ISO CAN FD protocol, a 3-bit counter and a parity bit are introduced in ISO CAN FD protocol. Thus the failure detection capability is improved for ISO CAN FD.

CAN FD mode supports both CAN classical frames and CAN FD frames. The FDF bit (the reserved bit in CAN classical frames) is used to distinguish between CAN classical and CAN FD format frames. When the FDF bit is recessive ‘1’, it is recognized as a CAN FD frame; otherwise, it is a classical frame. Compared with CAN classical frame, CAN FD frame does not support Rx FIFO, Rx FIFO DMA, and Pretended Networking function.

To enable CAN FD mode, set FDEN bit in CAN_CTL0 register to 1.

#### CAN FD BRS

In CAN FD mode, the data byte length is allowed up to 64 bytes for a CAN FD frame, and the bit time can switch to a higher speed of 8 Mbit/s for the Data Phase (from BRS bit to the first sample point of CRC Delimiter or to the starting of an error frame when an error condition is detected) of a CAN FD frame with BRS bit set (refer to ISO11898-1 or Bosch CAN FD Specification V1.0).

When BRSEN bit in CAN_FDCTL register is set to 1 (takes effect at the next message), and BRS bit in Tx mailbox is written as recessive ‘1’, then higher bit time (called as data bit time) will be used for the Data Phase of CAN FD frame, the nominal bit time will be used for the rest of the bits. The bit time is changed at the sample point of the BRS bit. The data bit time is configured in CAN_FDBT register. The nominal bit time is configured in CAN_BT register.

When BRSEN bit in CAN_FDCTL register is set to 0, or when BRS bit in Tx mailbox is written as 0, then nominal bit time will be used for the entire CAN FD frame.

**Note:** The length of time quantum should be the same for the entire CAN FD frame, to reduce the possibilities of phase error frames on the CAN bus.

In FD frames, all nodes shall accept an up to two bit long dominant phase of overlapping ACK

slot bits as a valid ACK, to compensate for phase shifts between the receivers. (Refer to ISO11898-1 specification)

### CAN FD ESI

The transmission of ESI bit (the bit before DLC bits, refer to ISO11898-1 or Bosch CAN FD Specification V1.0) is defined by ESI field in MDES0 word of Tx mailbox and ERRSI[1:0] bits in CAN_ERR1 register. If ESI field in MDES0 is 0, it will transmit the dominant bit by error active nodes and transmit the recessive bit by error passive nodes according to ERRSI[1:0] bits in CAN_ERR1 register. If ESI field in MDES0 is 1, it will transmit ESI field in MDES0 word.

### CAN FD CRC

Different CRC polynomials are used for different frame formats, results in a Hamming distance of 6:

- The CRC_15 polynomial is used for frames in CAN classical format: 0xC599
  $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$
- The CRC_17 polynomial is used for frames in CAN FD format with DATA field no more than 16 bytes: 0x3685B
  $x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x^3 + x^1 + 1$
- The CRC_21 polynomial is used for frames in CAN FD format with DATA field more than 16 bytes: 0x302899
  $x^{21} + x^{20} + x^{13} + x^{11} + x^7 + x^4 + x^3 + 1$

For transmission, these three types of CRC will all be calculated at the start of the frame, and the final CRC to be transmitted is determined by the FDF field and DLC field of the frame. After a successful transmission, when corresponding MSx bit of CAN_STAT register is set, the CAN_CRCCFD register is updated at the same time, with the calculated CRC for both CAN FD and non-FD messages. The CAN_CRCC register only stores the calculated CRC for CAN classical format frames.

For reception, the CRC polynomial used for CRC check is determined by the received FDF and DLC field.

**Note:** In Classical frames, the CRC delimiter is one single recessive bit. In FD frames, the CRC delimiter may consist of one or two recessive bits. A transmitter shall send only one recessive bit as CRC delimiter, but it shall accept two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. A receiver will send its acknowledge bit after the first CRC delimiter bit. Refer to ISO11898-1 specification.

### Bit stuff

The bit stuffing in CAN FD format frames is different from that in CAN classical format frames.

For transmission of CAN FD format frames, a fixed stuff bit is inserted before the first bit of the CRC field (regardless of the bit stuff conditions), and other fixed stuff bits are inserted after each 4 bits of the CRC field (fixed stuff bits are not included). The value of these fixed

stuff bits are the inverse value of their preceding bit. Refer to ISO11898-1 specification.

For the reception of CAN FD frames, these fixed stuffing bits will be ignored. When the value of the fixed stuff bit is the same as the value of its preceding bit, a stuff error is detected.

**Note:** For CAN FD format frames, fixed stuff bits are included in CRC calculation. For CAN classical format frames, stuff bits are not included.

### Resynchronization

Resynchronization and hard synchronization occur in CAN FD frames in the same way as in CAN classical frames. A node transmitting an CAN FD frame shall not resynchronize while it transmits the data phase of that frame.

### Transmitter delay compensation

The transmitter delay compensation is used for the data phase of CAN FD frames with BRS set, for the reason that in CAN FD frames with BRS bit set, the length of the CAN bit time in the data phase is shorter than the transmitter delay, thus the bit error check is influenced, the transmitter cannot receive its own transmitted bit latest at the sample point of that bit. The transmitter delay is measured from the falling edge of FDF bit of transmitted frame to the falling edge of FDF bit of received frame, shown in ***Figure 42-2. Transmitter delay***.

The transmitter delay compensation mechanism defines a secondary sample point SSP. When it is used, the transmitter shall ignore bit errors detected at the sample point. When TDCEN bit in CAN_FDCTL register is 1, this feature is enabled, then the bit check will be done between the actually received bit and the delayed transmitted bit (the delay is calculated based on the measured transmitter delay).

The transmitter delay compensation value is calculated in the equation follows:

$$t_{compensation} = t_{measure} + t_{offset} \quad (42\text{-}1)$$

with

$$t_{offset} = TDCO[4:0] \times t_{CANCLK} \quad (42\text{-}2)$$

$$t'_{offset} = t_{PBS1_FD} + t_{PTS_FD} + t_{SYNC_SEG} \quad (42\text{-}3)$$

$$t_{PBS1_FD} = (DPBS1[2:0] + 1) \times t_{q_FD} \quad (42\text{-}4)$$

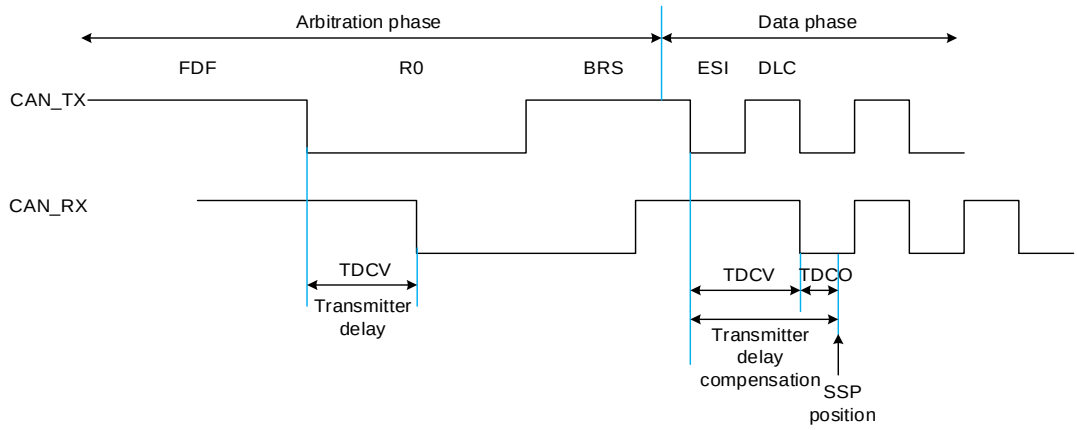
$$t_{PTS_FD} = DPTS[4:0] \times t_{q_FD} \quad (42\text{-}5)$$

$$t_{q_FD} = (DBAUDPSC[9:0] + 1) \times t_{CANCLK} \quad (42\text{-}6)$$

where the $t_{measure}$ is the measured transmitter delay, $t_{offset}$ is the transmitter delay compensation offset which is saved in the TDCO[4:0] bits of CAN_FDCTL register in unit of $t_{CANCLK}$, $t_{offset}$ should not be larger than the CAN data bit time. $t_{compensation}$ is the transmitter delay compensation value saved in TDCV[5:0] bits of CAN_FDCTL register in unit of $t_{CANCLK}$.

In the equations, the DPBS1[2:0] bits, DPTS[4:0] bits, DBAUDPSC[9:0] bits are all configured in CAN_FDBT register.

**Figure 42-2. Transmitter delay**




The maximum $t_{compensation}$ is $(3 \times \text{data bit time} - 2 \times t_{q_FD})$. When exceed this value, it is unable to compensate the transmitter delay, then TDCS bit in CAN_FDCTL register will be set. The implementation shall be able to compensate transmitter delays of at least two data bit times.

## 42.3.9. Errors and states

Transmit Error Counter (TECNT[7:0] bits in CAN_ERR0 register) and Receive Error Counter (RECNT[7:0] bits in the CAN_ERR0 register) take into account all errors in both CAN FD and non-FD messages, which get incremented or decremented according to the error condition. For detailed information about TECNT[7:0] and RECNT[7:0] management, please refer to the CAN standard.

For CAN FD format frames, a Receive Error Counter for data phase of CAN FD messages (REFCNT[7:0] bits in the CAN_ERR0 register) and a Transmit Error Counter for data phase of CAN FD messages (TEFCNT[7:0] bits in the CAN_ERR0 register) are used additionally only when the BRS field of the frame is set. They stop counting and keep their values when in Bus off state, and they restart counting from 0 after returned to error active state by Bus off recovery.

**Note:** When in Pretended Networking mode, receive error counters keep counting and error flags are saved, transmit error counters stop counting and save their values. When returns to normal mode, the CAN_ERR0 and CAN_ERR1 registers will be updated with the counter value and saved error flags.

### States

#### Error Passive State

If the value of TECNT[7:0] or RECNT[7:0] in CAN_ERR0 register increments to greater than 127, ERRSI[1:0] in CAN_ERR1 register is updated to 1 (error passive state).

**Error Active state**

If the node is in Error Passive state, and the value of either TECNT[7:0] or RECNT[7:0] in CAN_ERR0 register decrements to less than or equal to 127 when the other already satisfies this condition, ERRSI[1:0] in CAN_ERR1 register is updated to 0 (error active state).

**Bus off state**

If the value of TECNT[7:0] in CAN_ERR0 register becomes greater than 255, ERRSI[1:0] in CAN_ERR1 register is updated to 0b1x (Bus off state), and BOF bit in CAN_ERR1 register will be set, when BOIE bit in CAN_CTL1 register is set, an interrupt will be generated. The value of TECNT[7:0] will be reset to 0.

Bus off recovery:

To exit from Bus off state, the CAN has to wait for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX). When TECNT[7:0] in CAN_ERR0 register reaches 128, ERRSI[1:0] in CAN_ERR1 register is updated to 0 (error active state) and both TECNT[7:0] and RECNT[7:0] in CAN_ERR0 register are reset to 0.

Depending on ABORDIS bit in CAN_CTL1 register, CAN will recover from Bus off automatically or remain in Bus off state.

When ABORDIS is 0, enable automatic bus off recovery, CAN wil recover from Bus off automatically after the recovery sequence. If the ABORDIS is changed to 0 after the recovery sequence, then CAN will resynchronize to the bus by detecting 11 consecutive recessive bits.

When ABORDIS is 1, not enable automatic bus off recovery. If the ABORDIS is changed to 1 after the CAN entered Bus off state, automatic bus off recovery will be disabled at the next time the CAN entered Bus off state.

**Bus integration state**

If the node starts the protocol operation during Bus off recovery, or detects the protocol exception event (the event occurs when FDEN bit in CAN_CTL0 register is set to 0, and a FDF bit of a FD frame is received), the node enters the bus integration state. In this state, the synchronicity to the CAN bus is lost. CAN node can leave the bus integration state when the bus idle condition (the sequence of 11 consecutive recessive bits) is detected. Refer to the CAN Protocol standard (ISO 11898-1).

The protocol exception detection is controlled by PREEN bit in CAN_CTL2 register.

The edge filtering can be configured by EFDIS bit in CAN_CTL2 register, which is used during the bus integration state. When the edge filtering is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting for bus idle condition (the sequence of 11 consecutive recessive bits) is restarted. When edge filtering is performed, dominant bus-states shorter than a nominal bit time (the bits in data phase of a FD frame) will be ignored, stopped from being mistaken for an idle condition. Refer to the CAN Protocol standard (ISO

11898-1).

**Note:** Recommend to always enable edge filtering by reset EFDIS bit, to avoid mistakenly detection of bus idle condition.

#### Errors

If at least one of the error flags (ACKERR, BRERR, BDERR, CRCERR, FMERR, and STFERR bit in CAN_ERR1 register) is set, ERRSF bit in CAN_ERR1 register will be set. If ERRSIE bit in CAN_CTL1 register is set, an error interrupt will be generated.

If at least one of the error flags (BRFERR, BDFERR, CRCFERR, FMFERR, and STFFERR bit in CAN_ERR1 register) is set, ERRFSF bit in CAN_ERR1 register will be set. If ERRFSIE bit in CAN_CTL2 register is set, an error interrupt will be generated for errors detected in CAN FD frame data phase with BRS bit set.

**Acknowledge error**

If there is only one node operating, then it will lead to TECNT[7:0] in CAN_ERR0 register incrementing (to 128 at most by acknowledge error) in each message transmission, and an acknowledge error will occur, which is indicated by ACKERR bit in the CAN_ERR1 register.

**Bit recessive error**

When at least one bit sent as recessive ‘1’ is received as dominant ‘0’, a bit recessive error occurs. Refers to BRFERR and BRERR bit in CAN_ERR1 register.

**Bit dominant error**

When at least one bit sent as dominant ‘0’ is received as recessive ‘1’, a bit dominant error occurs. Refers to BDFERR and BDERR bit in CAN_ERR1 register.

**CRC error**

When the calculated CRC is different from the received CRC field of the frame, a CRC error occurs. Refers to CRCFERR and CRCERR bit in CAN_ERR1 register.

**Form error**

When a fixed-form bit field contains at least one illegal bit, a form error occurs. Refers to FMFERR and FMERR bit in CAN_ERR1 register.

**Stuff error**

Refers to STFFERR and STFERR bit in CAN_ERR1 register.

### 42.3.10. Communication parameters

#### Bit time

The CAN bit time from the CAN protocol has three segments as follows:

**Synchronization segment (SYNC_SEG)**: A bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_q$).

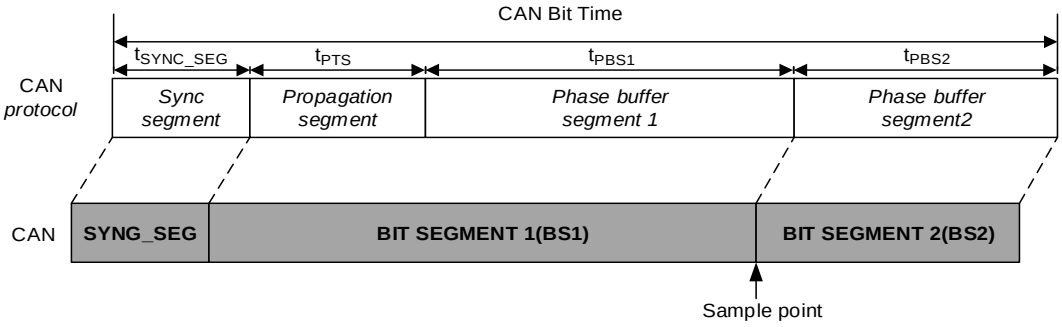
**Bit segment 1 (BS1)**: It defines the location of the sample point. It includes the Propagation segment and Phase buffer segment 1 in the CAN standard. It may be automatically lengthened to compensate for positive phase drifts due to different frequency of the various nodes of the network.

**Bit segment 2 (BS2)**: It defines the location of the sample point. It may also be automatically shortened to compensate for negative phase drifts. Its duration should programmed no less than 2 time quanta.

**Note:** The bit time configuration ranges must be in compliance with the CAN Protocol standard (ISO 11898-1).

CAN bit time is shown as in the ***Figure 42-3. CAN bit time***.

**Figure 42-3. CAN bit time**




**Synchronization Jump Width (SJW):** It can be lengthened or shortened to compensate for the Synchronization error of the CAN network node. It is configured by SJW[4:0] bits in CAN_BT register for nomainal bit time, or configured by DSJW[2:0] bits in CAN_FDBT register for data bit time.

The valid edge transition is defined as the first transition from a recessive bit to a dominant bit within a bit time, provided that the bus state detected at the previous sample point was recessive.

If a valid edge is detected in BS1, not in SYNC_SEG, BS1 is added with up to SJW maximumly, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2, not in SYNC_SEG, BS2 is cut down with up to SJW maximumly, so that the transmit point is moved earlier.

**Note:** For detailed explanations of hard synchronization and resynchronization, please refer to the ISO 11898 standard.

### Bit sampling

BSPMOD in CAN_CTL1 register defines the sampling mode of CAN bits at the Rx input pin.

When BSPMOD is 0, only one sample (the sample point) is used.

When BSPMOD is 1, three samples are used to determine the received bit value, that is the one on the sample point, and the two preceding samples.

**Note:** This bit cannot be set when CAN FD is enabled.

### Baudrate

CAN module has two clock domains:

- The clock of Control Interface and CAN registers derives from the APB2 clock.
- The clock of Protocol controller (CANCLK) can be configured by CANxSEL[1:0] bit in RCU_CFG2 register, to derive from oscillator clock, or APB2 clock, or APB2 clock devided by 2, or IRC64MDIV internal clock.

The CAN calculates its baudrate as follows:

$$\text{BaudRate} = \frac{1}{\text{CAN Bit Time}} \tag{42-7}$$

$$\text{CAN Bit Time} = t_{SYNC_SEG} + t_{PTS} + t_{PBS1} + t_{PBS2} \tag{42-8}$$

with

$$t_{SYNC_SEG} = 1 \times t_q \tag{42-9}$$

$$t_{PTS} = (N_{PTS} + 1) \times t_q \text{ or } t_{PTS} = N_{DPTS} \times t_q \tag{42-10}$$

$$t_{PBS1} = (N_{PBS1} + 1) \times t_q \tag{42-11}$$

$$t_{PBS2} = (N_{PBS2} + 1) \times t_q \tag{42-12}$$

$$t_q = (N_{BAUDPSC} + 1) \times t_{CANCLK} \tag{42-13}$$

In the equations, for nominal bit rate:

$N_{PTS}$, $N_{PBS1}$, $N_{PBS2}$, and $N_{BAUDPSC}$ are configured by the PTS[5:0] bits, PBS1[4:0] bits, PBS2[4:0] bits, and BAUDPSC[9:0] bits respectively in CAN_BT register.

For data bit rate:

$N_{DPTS}$, $N_{PBS1}$, $N_{PBS2}$, and $N_{BAUDPSC}$ are configured by the DPTS[4:0] bits, DPBS1[2:0] bits, DPBS2[2:0] bits, and DBAUDPSC[9:0] bits respectively in CAN_FDBT register.

### Timestamp

A 16-bit internal counter of the CAN hardware in CAN_TIMER register is used to generate the timestamp value. The value of the internal counter is sampled at SOF field on the CAN bus, and is written into the TIMESTAMP field of MDES0 or FDES0 word after a successful reception or transmission of a message.

The counter does not count in Inactive mode, or when LPS bit in CAN_CTL0 register is 1.

**Counter clock source**

When ITSRC bit in CAN_CTL2 register is 1, the TRIGSEL output CANx_EX_TIME_TICK is selected as the increasing condition of internal counter. The CANx_EX_TIME_TICK and APB2 bus clock are in the same clock domain, to ensure the internal counter be increased effectively, the pulse width of CANx_EX_TIME_TICK signal must be greater than or equal to the APB2 bus clock period.

When ITSRC bit in CAN_CTL2 register is 0, the CAN baudrate is selected as the increasing condition of internal counter, internal counter increments by one for each bit that is received or transmitted. When there is no message on the bus, it is counted with the baudrate programmed previously.

**Time synchronization**

If TSYNC bit in CAN_CTL1 register is 1, a SYNC message reception in the first mailbox descriptor will reset the internal counter for network time synchronization.

### 42.3.11. Interrupts

The CAN interrupt events and flags are list in ***Table 42-11. Interrupt events***.

**Table 42-11. Interrupt events**

| Interrupt event | | Flag | | | Enable control | | | |
|---|---|---|---|---|---|---|---|---|
| | | Bit | | Register | Enable bit | Control bit | Enable register | Control register |
| Bus off | | BOF | | CAN_ERR1 | BOIE | | CAN_CTL1 | |
| Bus off recovery | | BORF | | | BORIE | | CAN_CTL2 | |
| Error summary | Bit recessive error | ERRSF | BRERR | | ERRSIE | | CAN_CTL1 | |
| | Bit dominant error | | BDERR | | | | | |
| | ACK error | | ACKERR | | | | | |
| | CRC error | | CRCERR | | | | | |
| | Form error | | FMERR | | | | | |
| | Stuff error | | STFERR | | | | | |
| Error summary for FD frames with data bit time | Bit recessive error | ERRFSF | BRFERR | | ERRFSIE | | CAN_CTL2 | |
| | Bit dominant error | | BDFERR | | | | | |
| | CRC error | | CRCFERR | | | | | |
| | Form error | | FMFERR | | | | | |
| | Stuff error | | STFFERR | | | | | |
| Tx error warning | | TWERRIF | | | TWERRIE | WERREN | CAN_CTL1 | CAN_CTL0 |
| Rx error warning | | RWERRIF | | | RWERRIE | | | |
| Wakeup match | | WMS | | CAN_PN_STAT | WMIE | | CAN_PN_CTL0 | |
| Wakeup timeout | | WTOS | | | WTOIE | | | |
| Mailbox successful transmission or reception | | All bits | | CAN_STAT | All bits | RFEN = 0 | CAN_INTEN | CAN_CTL0 |
| | | MSx | | | MIEx | RFEN = 1 | | |
| Rx FIFO not empty | | MS5_RFNE | | | MIE5 | RFEN = 1 & DMAEN = 0 | | |
| Rx FIFO warning | | MS6_RFW | | | MIE6 | | | |
| Rx FIFO overflow | | MS7_RFO | | | MIE7 | | | |

## 42.4. Example for a typical configuration flow of CAN

After power-on reset or system reset, the following operation flow is a typical process for application to configure and run CAN:

- Configure CAN module clock source CANCLK, and enable CAN clock
  Configure CANxSEL[1:0] bits in CAN_CFG2 register to select the CAN module clock source. Program the RCU_APB2EN register to enable the CAN module clock.
- Setup the communication interface
  Configure GPIO and AFIO module to select PADs to alternate functions.
- Enter Inactive mode
  Because INAMOD bit, HALT bit, NRDY bit and INAS bit are default set after power-on reset or system reset, so CAN will automatically enters Inactive mode for configuration

of CAN registers.

- Service the flags in CAN_STAT register
  Read the Rx mailbox or Rx FIFO descriptor contents, clear the corresponding asserted flag bit in CAN_STAT register, then read the CAN_TIMER register at last for a complete flag bit service. If Rx FIFO is enabled, do a clearing FIFO operation by setting MS0 bit in CAN_STAT register to 1. Also clear the asserted flags by Tx mailboxes.
- Initialize the physical memory space for mailbox and Rx FIFO descriptors
  Configure memory space for mailbox and Rx FIFO descriptors totally by MSZ[4:0] bits in CAN_CTL0 register.
- Configure the communication parameters
  1) Configure the CAN nominal bit rate by PTS[5:0] bits, PBS1[4:0] bits, PBS2[4:0] bits, SJW[4:0] bits and BAUDPSC[9:0] bits in CAN_BT register.
  2) Configure bit sampling mode by BSPMOD bit in CAN_CTL1 register if needed.
  3) Configure PREEN bit and EFDIS bit for bus integration state if needed.
- Configure the control parameters for transmission
  1) Configure arbitration priority by MTO bit in CAN_CTL1 register and LAPRIOEN bit in CAN_CTL0 register.
  2) Configure arbitration start delay by ASD[4:0] bits of CAN_CTL2 register if needed.
  3) Enable transmission abort function for Tx mailbox descriptor configuration by MST bit in CAN_CTL0 register.
- Configure the control parameters for reception
  1) Choose whether use Rx FIFO and Rx FIFO DMA for reception or not by RFEN bit and DMAEN bit in CAN_CTL0 register.
  2) Configure Rx private filter & Rx mailbox queue feature by RPFQEN bit in CAN_CTL0 register.
  3) Configure receive filter related parameters by RFO bit, RRFRMS bit and IDERTR_RMF bit of CAN_CTL2 register.
  4) Configure filter data of the Rx mailbox and Rx FIFO by CAN_RMPUBF, CAN_RFIFOPUBF and CAN_RFIFOMPFx (x = 0..31) registers. If Rx FIFO is enabled, configure Rx FIFO ID filter table element format by FS[1:0] bits of CAN_CTL0 register, configure Rx FIFO ID filter table element number by RFFN[3:0] bits of CAN_CTL2 register.
- If CAN FD operation is needed
  1) Select CAN FD operation protocol by ISO bit in CAN_CTL2 regitser.
  2) Enable CAN FD operation by FDEN bit in CAN_CTL0 register.
  3) Initialize the mailbox data size by MDSZ[1:0] bits of CAN_FDCTL register.
  4) Configure CAN FD related transmitter delay compensation feature by TDCEN bit and TDCO[4:0] bits of CAN_FDCTL register if needed.
  5) Configure the CAN data bit rate by DPTS[4:0] bits, DPBS1[2:0] bits, DPBS2[2:0] bits DSJW[2:0] bits and DBAUDPSC[9:0] bits in CAN_FDBT register.
- Configure interrupts
  Enable the needed interrupts in CAN_CTL0, CAN_CTL1, CAN_CTL2 and CAN_INTEN registers.
- Initialize the Tx / Rx mailbox descriptors

1) If message transmission is needed, initialize the Tx mailbox descriptors.

2) If message reception is needed, initialize the Rx mailbox descriptors, if Rx FIFO is enabled, also initialize the Rx FIFO descriptors including the ID filter table elements.

- If Pretended Networking mode is required, set PNEN bit and PNMOD bit in CAN_CTL0 register and configure the necessary registers for wakeup.
- Exit Inactive mode

  Clear HALT bit in CAN_CTL0 register to exit Inactive mode, and CAN starts to synchronize to the CAN bus.

## 42.5. CAN registers

CAN0 base address: 0x4001 A000

CAN1 base address: 0x4001 B000

CAN2 base address: 0x4001 C000

### 42.5.1. Control register 0 (CAN_CTL0)

Address offset: 0x00
Reset value: 0x5900 000F

All bits except bit 30, 28, 25, 19 of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

All bits except bit 31, 27, 24, 20 of this register will be reset by software reset bit SWRST in CAN_CTL0 register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CANDIS | INAMOD | RFEN | HALT | NRDY | Reserved | SWRST | INAS | Reserved | | WERREN | LPS | PNEN | PNS | SRDIS | RPFQEN |
| rw | rw | rw | rw | r | | rw | r | | | rw | r | rw | r | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DMAEN | PNMOD | LAPRIOEN | MST | FDEN | Reserved | FS[1:0] | | Reserved | | | MSZ[4:0] | | | | |
| rw | rw | rw | rw | rw | | rw | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CANDIS | CAN disable<br>This bit is not affected by software reset bit SWRST in CAN_CTL0 register.<br>0: Enable CAN module<br>1: Disable CAN module |
| 30 | INAMOD | Inactive mode enable<br>0: Disable Inactive mode<br>1: Enable Inactive mode |
| 29 | RFEN | Rx FIFO enable<br>0: Disable Rx FIFO<br>1: Enable Rx FIFO |
| 28 | HALT | Halt CAN<br>0: No enter Inactive mode request<br>1: Enter Inactive mode if the INAMOD bit in CAN_CTL0 register is set |
| 27 | NRDY | Not ready<br>This bit indicates the state of whether the Protocol controller clock is disabled or |

| | | |
|---|---|---|
| | | not. When in Inactive mode, or in CAN_Disable mode, the Protocol controller clock is disabled, and CAN is not ready.<br>0: CAN is ready<br>1: CAN is not ready |
| 26 | Reserved | Must be kept at reset value. |
| 25 | SWRST | Software reset<br>When this bit is set, CAN internal state machines and CAN registers will be reset. This bit is automitically cleared by hardware when software reset is completed. Software reset has no effect when LPS bit in CAN_CTL0 register is set.<br>0: No effect<br>1: Software reset request |
| 24 | INAS | Inactive mode state<br>0: Not in Inactive mode<br>1: In Inactive mode |
| 23:22 | Reserved | Must be kept at reset value. |
| 21 | WERREN | Error warning enable<br>When this bit is set, the warning interrupt flag TWERRIF and RWERRIF bit in CAN_ERR1 register will be enabled to reflect the state change of TWERRF and RWERRF bit in CAN_ERR1 register respectively.<br>0: Disable Tx and Rx error warining<br>1: Enable Tx and Rx error warining |
| 20 | LPS | Low power state<br>0: Not in low power state<br>1: In low power state |
| 19 | PNEN | Pretended Networking mode enable<br>0: Disable Pretended Networking mode<br>1: Enable Pretended Networking mode |
| 18 | PNS | Pretended Networking state<br>0: Not in Pretended Networking state<br>1: In Pretended Networking state |
| 17 | SRDIS | Self reception disable<br>0: Enable self reception<br>1: Disable self reception |
| 16 | RPFQEN | Rx private filters enable & Rx mailbox queue enable<br>0: Disable Rx private filters & disable Rx mailbox queue<br>1: Enable Rx private filters & enable Rx mailbox queue |
| 15 | DMAEN | DMA enable<br>0: DMA feature for RX FIFO disabled. |

| | | |
|---|---|---|
| | | 1: DMA feature for RX FIFO enabled. |
| 14 | PNMOD | Pretended Networking mode selection<br>0: Not select Pretended Networking mode<br>1: Select Pretended Networking mode |
| 13 | LAPRIOEN | Local arbitration priority enable<br>0: Disable local arbitration priority<br>1: Enable local arbitration priority |
| 12 | MST | Mailbox stop transmission<br>0: Disable transmission abort<br>1: Enable transmission abort |
| 11 | FDEN | CAN FD operation enable<br>0: Disable CAN FD operation<br>1: Enable CAN FD operation |
| 10 | Reserved | Must be kept at reset value. |
| 9:8 | FS[1:0] | Format selection<br>This bit field defines the format of the Rx FIFO ID filter table elements.<br>00: Format A: One full ID (standard and extended) per ID filter table element<br>01: Format B: Two full standard IDs or two partial 14-bit extended IDs per ID filter table element<br>10: Format C: Four partial 8-bit IDs (standard and extended) per ID filter table element<br>11: Format D: All frames rejected |
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | MSZ[4:0] | Memory size<br>This bit field defines the maximum size of memory for message transmission and reception. The size is counted in uint of 4 words (equals to the size of a mailbox descriptor with 8-byte data), including mailbox and Rx FIFO.<br>Before configuring this bit field, the flags in CAN_STAT register must be serviced.<br>00000: 1 unit<br>00001: 2 units<br>…<br>11111: 32 units |

### 42.5.2. Control register 1 (CAN_CTL1)

Address offset: 0x04
Reset value: 0x0000 0000

The bits 12, 7, 5, 4, 3 of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

All bits of this register are not affected by software reset bit SWRST in CAN_CTL0 register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BOIE | ERRSIE | Reserved | LSCMOD | TWERRIE | RWERRIE | Reserved | | BSPMOD | ABORDIS | TSYNC | MTO | MMOD | Reserved | | |
| rw | rw | | rw | rw | rw | | | rw | rw | rw | rw | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15 | BOIE | Bus off interrupt enable<br>0: Disable Bus off interrupt<br>1: Enable Bus off interrupt |
| 14 | ERRSIE | Error summary interrupt enable<br>0: Disable error summary interrupt<br>1: Enable error summary interrupt |
| 13 | Reserved | Must be kept at reset value. |
| 12 | LSCMOD | Loopback and silent communication mode<br>0: Disable loopback and silent communication mode<br>1: Enable loopback and silent communication mode<br>**Note:** In this mode, SRDIS bit in CAN_CTL0 register, and TDCEN in CAN_FDCTL register cannot be set. |
| 11 | TWERRIE | Tx error warning interrupt enable<br>This bit can be written only when WERREN in CAN_CTL0 register is 1. This bit is read as zero when WERREN in CAN_CTL0 register is 0.<br>0: Disable Tx error warning interrupt<br>1: Enable Tx error warning interrupt |
| 10 | RWERRIE | Rx error warning interrupt enable<br>This bit can be written only when WERREN in CAN_CTL0 register is 1. This bit is read as zero when WERREN in CAN_CTL0 register is 0.<br>0: Disable Rx error warning interrupt<br>1: Enable Rx error warning interrupt |
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | BSPMOD | Bit sampling mode<br>0: One sample for the received bit<br>1: Three samples for the received bit |
| 6 | ABORDIS | Automatic Bus off recovery not enable<br>0: Enable automatic Bus off recovery |

| | | |
|---|---|---|
| | | 1: Not enable automatic Bus off recovery |
| 5 | TSYNC | Time synchronization enable<br>0: Disable time synchronization<br>1: Enable time synchronization |
| 4 | MTO | Mailbox transmission order<br>0: Highest priority mailbox is transmitted first<br>1: Lowest number mailbox is transmitted first |
| 3 | MMOD | Monitor mode<br>0: Disable Monitor mode<br>1: Enable Monitor mode |
| 2:0 | Reserved | Must be kept at reset value. |

### 42.5.3. Timer register (CAN_TIMER)

Address offset: 0x08
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CNT[15:0] | Counter value<br>This bit field contains the internal counter value used for timestamp generation. |

### 42.5.4. Receive mailbox public filter register (CAN_RMPUBF)

Address offset: 0x10
Reset value: 0xXXXX XXXX

This register is located in RAM.

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MFD31 | MFD30 | MFD29 | MFD28 | MFD27 | MFD26 | MFD25 | MFD24 | MFD23 | MFD22 | MFD21 | MFD20 | MFD19 | MFD18 | MFD17 | MFD16 |

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MFD15 | MFD14 | MFD13 | MFD12 | MFD11 | MFD10 | MFD9 | MFD8 | MFD7 | MFD6 | MFD5 | MFD4 | MFD3 | MFD2 | MFD1 | MFD0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MFDx | Mailbox filter data<br>MFD31 bit is used to filter the mailbox descriptor RTR field.<br>MFD30 bit is used to filter the mailbox descriptor IDE field.<br>MFDx (x = 0..28) bits are used to filter the mailbox descriptor ID field.<br>0: The bit is "don't care"<br>1: The bit is checked<br>**Note:** For standard frame, MDF18~MFD28 bits are use to filter the mailbox descriptor ID field. |

### 42.5.5. Error register 0 (CAN_ERR0)

Address offset: 0x1C
Reset value: 0x0000 0000

All bits of this register are read-only except in Inactive mode.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REFCNT[7:0] | | | | | | | | TEFCNT[7:0] | | | | | | | |
| rw0 | | | | | | | | rw0 | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RECNT[7:0] | | | | | | | | TECNT[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | REFCNT[7:0] | Receive error counter for data phase of FD frames with BRS bit set<br>This bit field can only be written as zero in Inactive mode. |
| 23:16 | TEFCNT[7:0] | Transmit error count for the data phase of FD frames with BRS bit set<br>This bit field can only be written as zero in Inactive mode. |
| 15:8 | RECNT[7:0] | Receive error count defined by the CAN standard |
| 7:0 | TECNT[7:0] | Transmit error count defined by the CAN standard |

### 42.5.6. Error register 1 (CAN_ERR1)

Address offset: 0x20
Reset value: 0x000X 000X

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRFERR | BDFERR | Reserved | CRCFERR | FMFERR | STFFERR | Reserved | | | | ERROVR | ERRFSF | BORF | SYN | TWERRIF | RWERRIF |
| rc | rc | | rc | rc | rc | | | | | rc_w1 | rc_w1 | rc_w1 | r | rc_w1 | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRERR | BDERR | ACKERR | CRCERR | FMERR | STFERR | TWERRF | RWERRF | IDLEF | TS | ERRSI[1:0] | | RS | BOF | ERRSF | Reserved |
| rc | rc | rc | rc | rc | rc | r | r | r | r | r | | r | rc_w1 | rc_w1 | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | BRFERR | Bit recessive error in data phase of FD frames with the BRS bit set<br>0: No error occurrence<br>1: At least one bit sent as recessive is received as dominant |
| 30 | BDFERR | Bit dominant error in data phase of FD frames with the BRS bit set<br>0: No error occurrence<br>1: At least one bit sent as dominant is received as recessive |
| 29 | Reserved | Must be kept at reset value. |
| 28 | CRCFERR | CRC error in data phase of FD frames with the BRS bit set<br>0: No error occurrence<br>1: A CRC error occurred |
| 27 | FMFERR | Form error in data phase of FD frames with the BRS bit set<br>0: No error occurrence<br>1: A form error occurred |
| 26 | STFFERR | Stuff error in data phase of FD frames with the BRS bit set<br>0: No error occurrence<br>1: A stuff error occurred |
| 25:22 | Reserved | Must be kept at reset value. |
| 21 | ERROVR | Error overrun<br>This bit indicates that an error condition occurred when any error flag is already set.<br>0: Error overrun not occurred<br>1: Error overrun occurred |
| 20 | ERRFSF | Error summary flag for data phase of FD frames with BRS bit set<br>This bit is logical ORed by the following bits:<br>CAN_ERR1[31]: Bit recessive error<br>CAN_ERR1[30]: Bit dominant error<br>CAN_ERR1[28]: CRC error<br>CAN_ERR1[27]: Form error<br>CAN_ERR1[26]: Stuff error |
| 19 | BORF | Bus off recovery flag<br>This bit is set when the the recovery sequence specified in the CAN standard on the CAN bus is detected and CAN is ready to recovery from Bus off. |

| | | |
|---|---|---|
| | | 0: No event occurrence<br>1: Bus off recovery sequence event occurs |
| 18 | SYN | Synchronization flag<br>0: Not synchronized to the CAN bus<br>1: Synchronized to the CAN bus |
| 17 | TWERRIF | Tx error warning interrupt flag<br>This bit is not used during Bus off state.<br>0: No event occurrence<br>1: TWERRF bit in CAN_ERR1 register changes from 0 to 1 |
| 16 | RWERRIF | Rx error warning interrupt flag<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No event occurrence<br>1: RWERRF bit in CAN_ERR1 register changes from 0 to 1 |
| 15 | BRERR | Bit recessive error for all format frames<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No error occurrence<br>1: At least one bit sent as recessive is received as dominant |
| 14 | BDERR | Bit dominant error for all format frames<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No error occurrence<br>1: At least one bit sent as dominant is received as recessive |
| 13 | ACKERR | ACK error<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No error occurrence<br>1: An ACK error occurred |
| 12 | CRCERR | CRC error<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No error occurrence<br>1: A CRC error occurred |
| 11 | FMERR | Form error<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No error occurrence<br>1: A form error occurred |
| 10 | STFERR | Stuff error<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No error occurrence<br>1: A stuffing error occurred |
| 9 | TWERRF | Tx error warning flag<br>0: No event occurrence |

| | | |
|---|---|---|
| | | 1: TECNT[7:0] in CAN_ERR0 register is greater than or equal to 96 |
| 8 | RWERRF | Rx error warning flag<br>This bit is updated when exiting from Pretended Networking mode.<br>0: No event occurrence.<br>1: RECNT[7:0] in CAN_ERR0 register is greater than or equal to 96 |
| 7 | IDLEF | IDLE flag<br>0: No event occurrence<br>1: In Bus idle state |
| 6 | TS | Transmitting state<br>0: CAN is not working in transmitting state<br>1: CAN is working in transmitting state |
| 5:4 | ERRSI[1:0] | Error state indicator<br>When MMOD bit in CAN_CTL1 register and SWRST bit in CAN_CTL0 register are both set to 1, this bit will be reset for one CAN bit time, and then changes to 0b01 to reflect Monitor mode state.<br>00: Error active<br>01: Error passive<br>1x: Bus off |
| 3 | RS | Receiving state<br>0: CAN is not working in receiving state<br>1: CAN is working in receiving state |
| 2 | BOF | Bus off flag<br>0: No event occurrence<br>1: In Bus off state |
| 1 | ERRSF | Error summary flag<br>This bit is logical ORed by the following bits:<br>CAN_ERR1[15]: Bit recessive error<br>CAN_ERR1[14]: Bit dominant error<br>CAN_ERR1[13]: ACK error<br>CAN_ERR1[12]: CRC error<br>CAN_ERR1[11]: Form error<br>CAN_ERR1[10]: Stuff error |
| 0 | Reserved | Must be kept at reset value. |

### 42.5.7. Interrupt enable register (CAN_INTEN)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| MIE31 | MIE30 | MIE29 | MIE28 | MIE27 | MIE26 | MIE25 | MIE24 | MIE23 | MIE22 | MIE21 | MIE20 | MIE19 | MIE18 | MIE17 | MIE16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MIE15 | MIE14 | MIE13 | MIE12 | MIE11 | MIE10 | MIE9 | MIE8 | MIE7 | MIE6 | MIE5 | MIE4 | MIE3 | MIE2 | MIE1 | MIE0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MIEx | Message transmission and reception interrupt enable<br>When Rx FIFO is disabled, these bits are used for mailbox number x (refers to ***Mailbox number***) interrupt configuration.<br>When Rx FIFO is enabled, MIE5 to MIE7 are used for Rx FIFO interrupt configuration, and mailbox interruption configuration bits are the bits x that are the same with the mailbox number x (refers to ***Mailbox number***).<br>0: Disable the corresponding interrupt<br>1: Enable the corresponding interrupt |

## 42.5.8. Status register (CAN_STAT)

Address offset: 0x30

Reset value: 0x0000 0000

The bits 1 to 7 of this register will be cleared by configuration change of RFEN bit in CAN_CTL0 register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MS31 | MS30 | MS29 | MS28 | MS27 | MS26 | MS25 | MS24 | MS23 | MS22 | MS21 | MS20 | MS19 | MS18 | MS17 | MS16 |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MS15 | MS14 | MS13 | MS12 | MS11 | MS10 | MS9 | MS8 | MS7_RFO | MS6_RFW | MS5_RFNE | MS4_RES | MS3_RES | MS2_RES | MS1_RES | MS0_RFC |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | MSx | Mailbox x state<br>x is the mailbox number, refers to ***Mailbox number***.<br>0: No successful transmission or reception has occurred in the mailbox descriptor<br>1: A successful transmission or reception has occurred in the mailbox descriptor |
| 7 | MS7_RFO | Mailbox 7 state / Rx FIFO overflow<br>0: No successful transmission or reception has occurred in the mailbox descriptor 7 when Rx FIFO is disabled. / Rx FIFO is not overflow when Rx FIFO is enabled.<br>1: A successful transmission or reception has occurred in the mailbox descriptor 7 when Rx FIFO is disabled. / Rx FIFO is overflow when Rx FIFO is enabled. |
| 6 | MS6_RFW | Mailbox 6 state / Rx FIFO warning<br>0: No successful transmission or reception has occurred in the mailbox descriptor |

| | | |
|---|---|---|
| | | 6 when Rx FIFO is disabled. / Rx FIFO has no warning when Rx FIFO is enabled.<br>1: A successful transmission or reception has occurred in the mailbox descriptor 6 when Rx FIFO is disabled. / Rx FIFO almost full warning when Rx FIFO is enabled. |
| 5 | MS5_RFNE | Mailbox 5 state / Rx FIFO not empty<br>0: No successful transmission or reception has occurred in the mailbox descriptor 5 when Rx FIFO is disabled. / Rx FIFO is empty when Rx FIFO is enabled.<br>1: A successful transmission or reception has occurred in the mailbox descriptor 5 when Rx FIFO is disabled. / Rx FIFO is not empty when Rx FIFO is enabled. |
| 4 | MS4_RES | Mailbox 4 state / Reserved<br>Similar to MS1_RES description. |
| 3 | MS3_RES | Mailbox 3 state / Reserved<br>Similar to MS1_RES description. |
| 2 | MS2_RES | Mailbox 2 state / Reserved<br>Similar to MS1_RES description. |
| 1 | MS1_RES | Mailbox 1 state / Reserved<br>0: No successful transmission or reception has occurred in the mailbox descriptor 1 when Rx FIFO is disabled. / Reserved when Rx FIFO is enabled.<br>1: A successful transmission or reception has occurred in the mailbox descriptor 1 when Rx FIFO is disabled. / Reserved when Rx FIFO is enabled. |
| 0 | MS0_RFC | Mailbox 0 state / Clear Rx FIFO bit<br>0: No successful transmission or reception has occurred in the mailbox descriptor 0 when Rx FIFO is disabled. / No effect when Rx FIFO is enabled.<br>1: A successful transmission or reception has occurred in the mailbox descriptor 0 when Rx FIFO is disabled. / Clear Rx FIFO when Rx FIFO is enabled, only allowed to written in Inactive mode, refers to ***Clear FIFO***. |

### 42.5.9. Control register 2 (CAN_CTL2)

Address offset: 0x34
Reset value: 0x00A0 0000

All bits except bit 31, 30 of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

All bits of this register are not reset by software reset bit SWRST in CAN_CTL0 register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRFSIE | BORIE | Reserved | | RFFN[3:0] | | | | ASD[4:0] | | | | | RFO | RRFRMS | IDERTR_RMF |
| rw | rw | | | rw | | | | rw | | | | | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ITSRC | PREEN | Reserved | ISO | EFDIS | Reserved | | | | | | | | | | |
| rw | rw | | rw | rw | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | ERRFSIE | Error summary interrupt enable bit for data phase of FD frames with BRS bit set<br>0: Disable error summary interrupt for data phase of FD frames with BRS bit set<br>1: Enable error summary interrupt for data phase of FD frames with BRS bit set |
| 30 | BORIE | Bus off recovery interrupt enable<br>0: Disable Bus off recovery interrupt<br>1: Enable Bus off recovery interrupt |
| 29:28 | Reserved | Must be kept at reset value. |
| 27:24 | RFFN[3:0] | Rx FIFO filter number |

**Table 42-12. Rx FIFO filter element number**

| RFFN[3:0] | Rx FIFO filter element number | Rx FIFO occupied space | Available mailboxes |
|---|---|---|---|
| 0000 | 8 | Mailbox descriptor 0 - 7 | Mailbox 8 - 31 |
| 0001 | 16 | Mailbox descriptor 0 - 9 | Mailbox 10 - 31 |
| 0002 | 24 | Mailbox descriptor 0 - 11 | Mailbox 12 - 31 |
| 0003 | 32 | Mailbox descriptor 0 - 13 | Mailbox 14 - 31 |
| 0004 | 40 | Mailbox descriptor 0 - 15 | Mailbox 16 - 31 |
| 0005 | 48 | Mailbox descriptor 0 - 17 | Mailbox 18 - 31 |
| 0006 | 56 | Mailbox descriptor 0 – 19 | Mailbox 20 - 31 |
| 0007 | 64 | Mailbox descriptor 0 – 21 | Mailbox 22 - 31 |
| 0008 | 72 | Mailbox descriptor 0 – 23 | Mailbox 24 - 31 |
| 0009 | 80 | Mailbox descriptor 0 – 25 | Mailbox 26 - 31 |
| 000A | 88 | Mailbox descriptor 0 – 27 | Mailbox 28 - 31 |
| 000B | 96 | Mailbox descriptor 0 – 29 | Mailbox 30 - 31 |
| 000C | 104 | Mailbox descriptor 0 – 31 | none |
| others | 104 | Mailbox descriptor 0 - 31 | none |

This bit field must not be programmed with values that cause memory occupied by Rx FIFO to exceed the available memory size which is defined by MSZ[4:0] bits in CAN_CTL0 register, otherwise the exceeding ones will not be functional.

| | | |
|---|---|---|
| 23:19 | ASD[4:0] | Arbitration start delay<br>This bit field defines how many CAN bits the Tx arbitration process start point can be delayed. |
| 18 | RFO | Receive filter order<br>0: Rx FIFO is filtered first<br>1: Mailboxes are filtered first |
| 17 | RRFRMS | Remote request frame is stored |

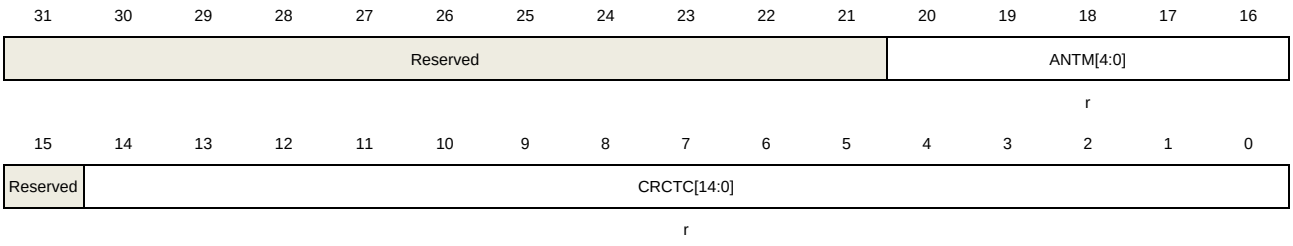
| | | |
|---|---|---|
| | | 0: When a mailbox with CODE=RANSWER and a matching ID is found, a remote answer frame is automatically generated.<br>1: When a mailbox with CODE=RANSWER and a matching ID is found, the remote request frame is stored as if it were a data frame, without automatically transmitting a remote answer frame. |
| 16 | IDERTR_RMF | IDE and RTR field filter type for Rx mailbox reception<br>This bit defines the matching of IDE and RTR field in Rx mailbox descriptor with the received bit.<br>0: IDE field is always compared, and RTR is never compared. Regardless of the filter data configurations in related filter register.<br>1: Filtering of IDE and RTR fields are enabled, by filter data configurations in related filter register. |
| 15 | ITSRC | Internal counter source<br>0: CAN baudrate<br>1: External trigger CANx_EX_TIME_TICK from TRIGSEL output |
| 14 | PREEN | Protocol exception detection enable by CAN standard<br>0: Disable protocol exception detection<br>1: Enable protocol exception detection |
| 13 | Reserved | Must be kept at reset value. |
| 12 | ISO | ISO CAN FD<br>0: Non-ISO CAN FD protocol operation is applied<br>1: ISO CAN FD protocol operation is applied |
| 11 | EFDIS | Edge filtering disable<br>0: Enable edge filtering<br>1: Disable edge filtering |
| 10:0 | Reserved | Must be kept at reset value. |

## 42.5.10. CRC for classical frame register (CAN_CRCC)

Address offset: 0x44

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:16 | ANTM[4:0] | Associated number of mailbox for transmitting the CRCTC[14:0] value<br>This bit field contains the number of the mailbox which transmits the CRCTC[14:0] value. |
| 15 | Reserved | Must be kept at reset value. |
| 14:0 | CRCTC[14:0] | Transmitted CRC value for classical frames<br>This bit field contains the CRC value of the last successfully transmitted message in classical format. |

### 42.5.11. Receive FIFO public filter register (CAN_RFIFOPUBF)

Address offset: 0x48
Reset value: 0xXXXX XXXX

This register is located in RAM.

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FFD31 | FFD30 | FFD29 | FFD27 | FFD27 | FFD26 | FFD25 | FFD24 | FFD23 | FFD22 | FFD21 | FFD20 | FFD19 | FFD18 | FFD17 | FFD16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FFD15 | FFD14 | FFD13 | FFD12 | FFD11 | FFD10 | FFD9 | FFD8 | FFD7 | FFD6 | FFD5 | FFD4 | FFD3 | FFD2 | FFD1 | FFD0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

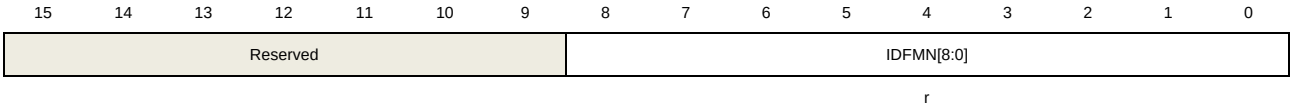
| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FFDx | Rx FIFO filter data<br>Each bit is used for filtering the corresponding ID filter table element bit, except the reserved bit in ID filter table element.<br>0: The bit is "don't care"<br>1: The bit is checked |

### 42.5.12. Receive FIFO identifier filter matching number register (CAN_RFIFOIFMN)

Address offset: 0x4C
Reset value: 0xXXXX XXXX

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |




| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8:0 | IDFMN[8:0] | Identifier filter matching number<br>This field is valid only when MS5_RFNE bit in CAN_STAT register is 1.<br>This bit field indicates which ID filter table element matches the received message that is in the output of the Rx FIFO. If more than one element is matched, the ID filter table element with the lowest number is stored. |

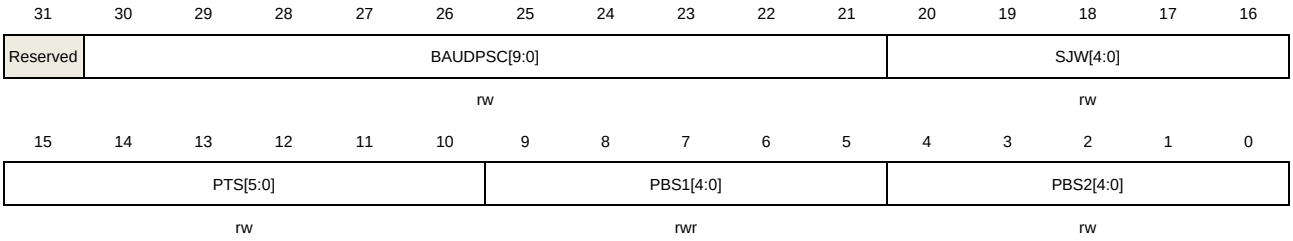
## 42.5.13. Bit timing register (CAN_BT)

Address offset: 0x50
Reset value: 0x0100 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register is not affected by software reset bit SWRST in CAN_CTL0 register.

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:21 | BAUDPSC[9:0] | Baud rate prescaler<br>The CAN baud rate prescaler = BAUDPSC[9:0] + 1. |
| 20:16 | SJW[4:0] | Resynchronization jump width<br>Resynchronization jump width time quantum = SJW[4:0] + 1 |
| 15:10 | PTS[5:0] | Propagation time segment<br>Propagation time segment time quantum = PTS[5:0] + 1 |
| 9:5 | PBS1[4:0] | Phase buffer segment 1<br>Phase buffer segment 1 time quantum = PBS1[4:0] + 1 |
| 4:0 | PBS2[4:0] | Phase buffer segment 2 |

Phase buffer segment 2 time quantum = PBS2[4:0] + 1

## 42.5.14. Receive FIFO/mailbox private filter x register (CAN_RFIFOMPFx)(x=0..31)

Address offset: 0x880 + 4 × x
Reset value: 0xXXXX XXXX

These register is located in RAM.

All bits of these registers should be configured in Inactive mode only, because they are blocked by hardware in other modes.

These registers are not affected by software reset bit SWRST in CAN_CTL0 register.

These registers have to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FMFD31 | FMFD30 | FMFD29 | FMFD27 | FMFD27 | FMFD26 | FMFD25 | FMFD24 | FMFD23 | FMFD22 | FMFD21 | FMFD20 | FMFD19 | FMFD18 | FMFD17 | FMFD16 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FMFD15 | FMFD14 | FMFD13 | FMFD12 | FMFD11 | FMFD10 | FMFD9 | FMFD8 | FMFD7 | FMFD6 | FMFD5 | FMFD4 | FMFD3 | FMFD2 | FMFD1 | FMFD0 |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | FMFDx | FIFO/mailbox filter data<br>If used as mailbox filters, refer to the MFDx bits in CAN_RMPUBF register.<br>If used as Rx FIFO filters, refer to the FFDx bits in CAN_RFIFOPUBF register.<br>0: The bit is "don't care"<br>1: The bit is checked |

## 42.5.15. Pretended Networking mode control register 0 (CAN_PN_CTL0)

Address offset: 0xB00
Reset value: 0x0000 0100

All bits except bit 17, 16 of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | WTOIE | WMIE |
| | | | | | | | | | | | | | | rw | rw |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| NMM[7:0] | | | | | | | | Reserved | | DATAFT[1:0] | | IDFT[1:0] | | FFT[1:0] | |
| rw | | | | | | | | | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | WTOIE | Wakeup timeout interrupt enable<br>0: Disable wakeup timeout interrupt<br>1: Enable wakeup timeout interrupt |
| 16 | WMIE | Wakeup match interrupt enable<br>0: Disable wakeup match interrupt<br>1: Enable wakeup match interrupt |
| 15:8 | NMM[7:0] | Number of messages matching times<br>An event counter is used in the wakeup message filter, in which a transistion on the output of event after N input matching events.<br>00000001: N = 1<br>00000010: N = 2<br>……<br>11111111: N = 255 |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | DATAFT[1:0] | DATA field filtering type in Pretended Networking mode<br>00: Only messages with DATA field equal to the expected data field through data filter are matched<br>01: Messages with DATA field greater than or equal to the expected data low threshold are matched<br>10: Messages with DATA field smaller than or equal to the expected data high threshold are matched<br>11: Messages with DATA field greater than or equal to the expected data low threshold, and smaller than or equal to the expected data high threshold are matched |
| 3:2 | IDFT[1:0] | ID field filtering type in Pretended Networking mode<br>00: Only messages with ID field equal to the expected identifier through identifier filter are matched<br>01: Messages with ID field greater than or equal to the expected identifier low threshold are matched<br>10: Messages with ID field smaller than or equal to the expected identifier high threshold are matched<br>11: Messages with ID field greater than or equal to the expected identifier low threshold, and smaller than or equal to the expected identifier high threshold are matched |
| 1:0 | FFT[1:0] | Frame filtering type in Pretended Networking mode<br>00: All fields except DATA field, DLC field are filtered<br>01: All fields are filtered<br>10: All fields except DATA field, DLC field are filtered with NMM[7:0] matching times<br>11: All fields are filtered with NMM[7:0] matching times |

### 42.5.16. Pretended Networking mode timeout register (CAN_PN_TO)

Address offset: 0xB04
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

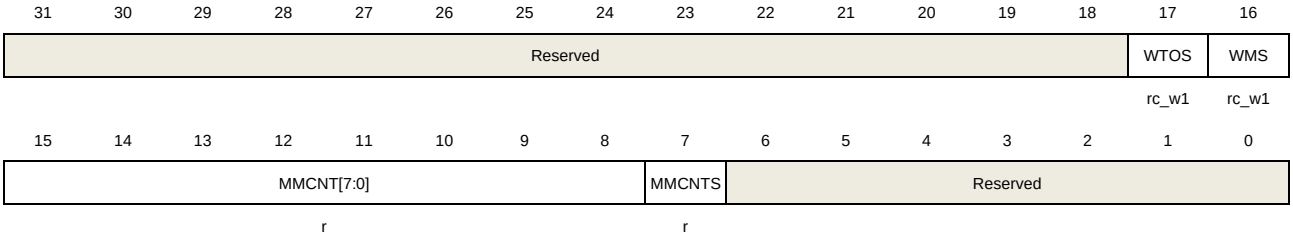



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | WTO[15:0] | Wakeup timeout<br>The timeout is counted by step of 64 times the CAN Bit Time. Wakeup timeout is default disabled. |

### 42.5.17. Pretended Networking mode status register (CAN_PN_STAT)

Address offset: 0xB08
Reset value: 0x0000 0080

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | WTOS | Wakeup timeout flag status<br>0: No wakeup timeout event occured<br>1: Wakeup timeout event occured |
| 16 | WMS | Wakeup match flag status<br>0: No wakeup match event occured<br>1: Wakeup match event occured |

| | | |
|---|---|---|
| 15:8 | MMCNT[7:0] | Matching message counter in Pretended Networking mode<br>This bit field indicates the matching message number during Pretended Networking mode. These bits are cleared when node enters Pretended Networking mode, they are not affected by software reset. |
| 7 | MMCNTS | Matching message counter state<br>This bit is set to 1 to show the value of MMCNT[7:0] is valid.<br>0: Matching message counter MMCNT[7:0] is updating<br>1: Matching message counter MMCNT[7:0] is valid |
| 6:0 | Reserved | Must be kept at reset value. |

### 42.5.18. Pretended Networking mode expected identifier 0 register (CAN_PN_EID0)

Address offset: 0xB0C
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | EIDE | ERTR | EID_ELT[28:16] | | | | | | | | | | | | |
| | rw | rw | rw | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EID_ELT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | EIDE | Expected IDE in Pretended Networking mode<br>0: Standard frame format<br>1: Extended frame format |
| 29 | ERTR | Expected RTR in Pretended Networking mode<br>0: Data frame<br>1: Remote frame |
| 28:0 | EIDF_ELT[28:0] | Expected ID field / expected ID low threshold in Pretended Networking mode<br>This bit field is used as expected ID field when IDFT[1:0] bit field in CAN_PN_CTL0 register is 0 / 1 / 2, or is used as expected ID low threshold when IDFT[1:0] bit field is 3.<br>For extended frame format, all 29 bits are used.<br>For standard frame format, bits 18 to 28 are used. |

### 42.5.19. Pretended Networking mode expected DLC register (CAN_PN_EDLC)

Address offset: 0xB10
Reset value: 0x0000 0008

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | DLCELT[3:0] | | | |
| | | | | | | | | | | | | rw | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | DLCEHT[3:0] | | | |
| | | | | | | | | | | | | rw | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:20 | Reserved | Must be kept at reset value. |
| 19:16 | DLCELT[3:0] | DLC expected low threshold in Pretended Networking mode |
| 15:4 | Reserved | Must be kept at reset value. |
| 3:0 | DLCEHT[3:0] | DLC expected high threshold in Pretended Networking mode |

### 42.5.20. Pretended Networking mode expected data low 0 register (CAN_PN_EDL0)

Address offset: 0xB14
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB0ELT[7:0] | | | | | | | | DB1ELT[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB2ELT[7:0] | | | | | | | | DB3ELT[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB0ELT[7:0] | Data byte 0 expected low threshold in Pretended Networking mode<br>Refer to DB3ELT[7:0] descriptions. |
| 23:16 | DB1ELT[7:0] | Data byte 1 expected low threshold in Pretended Networking mode |

| | | |
|---|---|---|
| | | Refer to DB3ELT[7:0] descriptions. |
| 15:8 | DB2ELT[7:0] | Data byte 2 expected low threshold in Pretended Networking mode<br>Refer to DB3ELT[7:0] descriptions. |
| 7:0 | DB3ELT[7:0] | Data byte 3 expected low threshold in Pretended Networking mode<br>This bit field is used as expected DATA field when DATAFT[1:0] bit field in CAN_PN_CTL0 register is 0 / 1 / 2, or is used as expected DATA low threshold when DATAFT[1:0] bit field is 3. |

### 42.5.21. Pretended Networking mode expected data low 1 register (CAN_PN_EDL1)

Address offset: 0xB18
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB4ELT[7:0] | | | | | | | | DB5ELT[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB6ELT[7:0] | | | | | | | | DB7ELT[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB4ELT[7:0] | Data byte 4 expected low threshold in Pretended Networking mode<br>Refer to DB3ELT[7:0] descriptions. |
| 23:16 | DB5ELT[7:0] | Data byte 5 expected low threshold in Pretended Networking mode<br>Refer to DB3ELT[7:0] descriptions. |
| 15:8 | DB6ELT[7:0] | Data byte 6 expected low threshold in Pretended Networking mode<br>Refer to DB3ELT[7:0] descriptions. |
| 7:0 | DB7ELT[7:0] | Data byte 7 expected low threshold in Pretended Networking mode<br>Refer to DB3ELT[7:0] descriptions. |

### 42.5.22. Pretended Networking mode identifier filter / expected identifier 1 register (CAN_PN_IFEID1)

Address offset: 0x B1C
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).



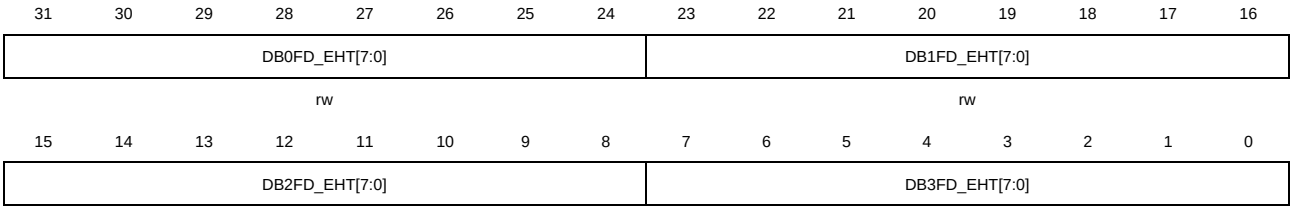

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | IDEFD | IDE filter data in Pretended Networking mode<br>0: The bit is "don't care"<br>1: The bit is checked |
| 29 | RTRFD | RTR filter data in Pretended Networking mode<br>0: The bit is "don't care"<br>1: The bit is checked |
| 28:0 | IDFD_EHT[28:0] | ID filter data / ID expected high threshold in Pretended Networking mode<br>**ID filter data** (when IDFT[1:0] bit field in CAN_PN_CTL0 register is 0):<br>0: The bit is "don't care"<br>1: The bit is checked<br>**ID expected high threshold** (when IDFT[1:0] bit field is 3).<br>**Bits reserved** (when IDFT[1:0] bit field is 1 or 2).<br>For extended frame format, all 29 bits are used.<br>For standard frame format, bits 18 to 28 are used. |

## 42.5.23. Pretended Networking mode data 0 filter / expected data high 0 register (CAN_PN_DF0EDH0)

Address offset: 0xB20
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).



rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB0FD_EHT[7:0] | Data byte 0 filter data / Data byte 0 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |
| 23:16 | DB1FD_EHT[7:0] | Data byte 1 filter data / Data byte 1 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |
| 15:8 | DB2FD_EHT[7:0] | Data byte 2 filter data / Data byte 2 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |
| 7:0 | DB3FD_EHT[7:0] | Data byte 3 filter data / Data byte 2 expected high threshold in Pretended Networking mode<br>**Data byte 3 filter data** (when DATAFT[1:0] bit field in CAN_PN_CTL0 register is 0):<br>0: The bit is "don't care"<br>1: The bit is checked<br>**Data byte 3 expected high threshold** (when DATAFT[1:0] bit field is 3).<br>**Bits reserved** (when DATAFT[1:0] bit field is 1 or 2). |

## 42.5.24. Pretended Networking mode data 1 filter / expected data high 1 register (CAN_PN_DF1EDH1)

Address offset: 0xB24
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB4FD_HTF[7:0] | | | | | | | | DB5FD_HTF[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DB6FD_HTF[7:0] | | | | | | | | DB7FD_HTF[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DB4FD_HTF[7:0] | Data byte 4 filter data / Data byte 4 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |

| | | |
|---|---|---|
| 23:16 | DB5FD_HTF[7:0] | Data byte 5 filter data / Data byte 5 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |
| 15:8 | DB6FD_HTF[7:0] | Data byte 6 filter data / Data byte 6 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |
| 7:0 | DB7FD_HTF[7:0] | Data byte 7 filter data / Data byte 7 expected high threshold in Pretended Networking mode<br>Refer to DB3FD_EHT[7:0] descriptions. |

### 42.5.25. Pretended Networking mode received wakeup mailbox x control status information register (CAN_PN_RWMxCS)(x=0..3)

Address offset: 0xB40 + 16 * x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | RSRR | RIDE | RRTR | RDLC[3:0] | | | |
| | | | | | | | | | r | r | r | r | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | | |

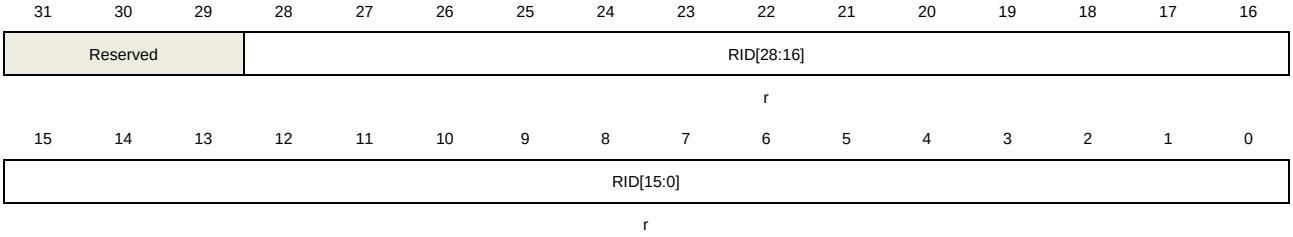
| Bits | Fields | Descriptions |
|---|---|---|
| 31:23 | Reserved | Must be kept at reset value. |
| 22 | RSRR | Received SRR bit |
| 21 | RIDE | Received IDE bit<br>0: Frame format is standard<br>1: Frame format is extended |
| 20 | RRTR | Received RTR bit<br>0: Frame is data frame<br>1: Frame is remote frame |
| 19:16 | RDLC[3:0] | Received DLC bits<br>The bit field indicates the valid data byte length. |
| 15:0 | Reserved | Must be kept at reset value. |

### 42.5.26. Pretended Networking mode received wakeup mailbox x identifier register (CAN_PN_RWMxI)(x=0..3)

Address offset: 0xB44 + 16 * x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).



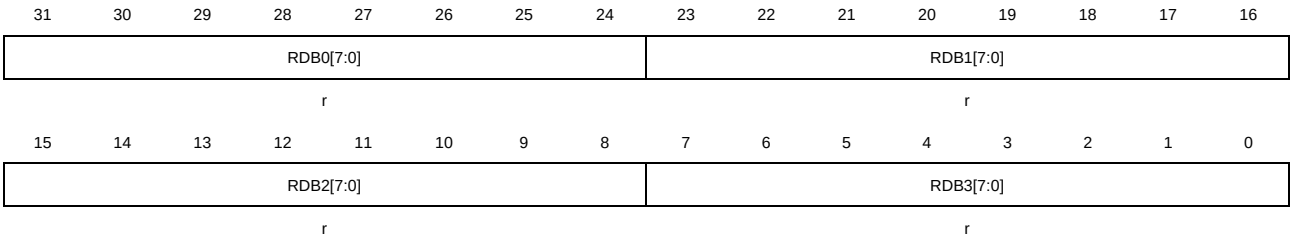

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:0 | RID[28:16] | Received ID bits<br>For extended frame format, all 29 bits are used for ID storage.<br>For standard frame format, bits 18 to 28 are used for ID storage. |

### 42.5.27. Pretended Networking mode received wakeup mailbox x data 0 register (CAN_PN_RWMxD0)(x=0..3)

Address offset: 0xB48 + 16 * x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | RDB0[7:0] | Received data byte 0 |
| 23:16 | RDB1[7:0] | Received data byte 1 |
| 15:8 | RDB2[7:0] | Received data byte 2 |
| 7:0 | RDB3[7:0] | Received data byte 3 |

### 42.5.28. Pretended Networking mode received wakeup mailbox x data 1 register (CAN_PN_RWMxD1)(x=0..3)

Address offset: 0xB4C + 16 * x
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDB4[7:0] | | | | | | | | RDB5[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RDB6[7:0] | | | | | | | | RDB7[7:0] | | | | | | | |
| r | | | | | | | | r | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | RDB4[7:0] | Received data byte 4 |
| 23:16 | RDB5[7:0] | Received data byte 5 |
| 15:8 | RDB6[7:0] | Received data byte 6 |
| 7:0 | RDB7[7:0] | Received data byte 7 |

### 42.5.29. FD control register (CAN_FDCTL)

Address offset: 0xC00
Reset value: 0x8000 0101

Bits 17:16, 15, 12:8 of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register is not affected by software reset bit SWRST in CAN_CTL0 register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRSEN | Reserved | | | | | | | | | | | | | MDSZ[1:0] | |
| rw | | | | | | | | | | | | | | rw | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TDCEN | TDCS | Reserved | TDCO[4:0] | | | | | Reserved | | TDCV[5:0] | | | | | |
| rw | rc_w1 | | rw | | | | | | | r | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | BRSEN | Bit rate of data switch enable<br>0: Bit rate not switch<br>1: The bit rate shall be switched from the nominal bit rate to the preconfigured data bit rate during the data phase when BRS bit in Tx mailbox is recessive ‘1’ |

| | | |
|---|---|---|
| 30:18 | Reserved | Must be kept at reset value. |
| 17:16 | MDSZ[1:0] | Mailbox data size<br>00: 8 bytes per mailbox<br>01: 16 bytes per mailbox<br>10: 32 bytes per mailbox<br>11: 64 bytes per mailbox |
| 15 | TDCEN | Transmitter delay compensation enable<br>**Note:** Transmitter delay compensation must be disabled when loopback and silent mode is enabled.<br>0: Transmitter delay compensation is disabled<br>1: Transmitter delay compensation is enabled |
| 14 | TDCS | Transmitter delay compensation status<br>When this bit is set, the transmitter delay is out of compensation range, it is unable to compensate the transmitter delay for bit check.<br>0: Transmitter delay is in compensation range<br>1: Transmitter delay is out of compensation range |
| 13 | Reserved | Must be kept at reset value. |
| 12:8 | TDCO[4:0] | Transmitter delay compensation offset<br>These bits are set to the transmitter delay compensation offset value which defines the distance between the measured delay from CANTX to CANRX and the second sample point for CAN FD frames with BRS bit set. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:0 | TDCV[5:0] | Transmitter delay compensation value<br>These bits are set by hardware to display the summary of the measured transmitter delay value and the transmitter delay compensation offset. |

## 42.5.30. FD bit timing register (CAN_FDBT)

Address offset: 0xC04
Reset value: 0x0000 0000

All bits of this register should be configured in Inactive mode only, because they are blocked by hardware in other modes.

This register is not affected by software reset bit SWRST in CAN_CTL0 register.

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | DBAUDPSC[9:0] | | | | | | | | | | Reserved | DSJW[2:0] | | |
| | | rw | | | | | | | | | | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | DPTS[4:0] | | | | | Reserved | | DPBS1[2:0] | | | Reserved | | DPBS2[2:0] | | |

rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29:20 | DBAUDPSC[9:0] | Baud rate prescaler for data bit time<br>The CAN data bit time baud rate prescaler = BAUDPSC[9:0] + 1. |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | DSJW[2:0] | Resynchronization jump width for data bit time<br>Resynchronization jump width time quantum = DSJW[2:0] + 1 |
| 15 | Reserved | Must be kept at reset value. |
| 14:10 | DPTS[4:0] | Propagation time segment for data bit time<br>Propagation time segment time quantum = DPTS[4:0] |
| 9:8 | Reserved | Must be kept at reset value. |
| 7:5 | DPBS1[2:0] | Phase buffer segment 1 for data bit time<br>Phase buffer segment 1 time quantum = DPBS1[2:0] + 1 |
| 4:3 | Reserved | Must be kept at reset value. |
| 2:0 | DPBS2[2:0] | Phase buffer segment 2 for data bit time<br>Phase buffer segment 2 time quantum = DPBS2[2:0] + 1 |

## 42.5.31. CRC for classical and FD frame register (CAN_CRCCFD)

Address offset: 0xC08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | ANTM[4:0] | | | | | Reserved | | | CRCTCI[20:16] | | | | |
| | | | r | | | | | | | | | | r | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRCTCI[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:24 | ANTM[4:0] | Associated number of mailbox for transmitting the CRCTCI[20:0] value<br>This bit field contains the number of the mailbox which transmits the CRCTCI[20:0] value for both classical and FD frames. |
| 23:21 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 20:0 | CRCTCI[20:0] | Transmitted CRC value for classical and ISO / non-ISO FD frames<br>For CRC_15, bits 0 to 14 are used, the other bits are zeros, and the value is the same as the value of CRCTC[14:0] in CAN_CRCC register.<br>For CRC_17, bits 0 to 16 are used, the other bits are zeros.<br>For CRC_21, all 21 bits are used. |

# 43. Ethernet (ENET) (For GD32H737_757_759)

## 43.1. Overview

This chapter describes the Ethernet peripheral module. There are two media access controller (MAC) designed in Ethernet module to support 10 / 100Mbps interface speed. For more efficient datas transfer between Ethernet and memory, a DMA controller is designed in this module. The support interface protocol for Ethernet is media independent interface (MII) and reduced media independent interface (RMII). This module is mainly compliant with the following two standards: IEEE 802.3-2002 and IEEE 1588-2008.

## 43.2. Characteristics

**MAC feature**

- 10Mbit / s and 100Mbit / s data transfer rates support.
- MII and RMII interface support.
- Loopback mode support for diagnosis.
- CSMA / CD Protocol for Half-duplex back-pressure operation support.
- IEEE 802.3x flow control protocol support. Automatic delay a pause time which is decoded from a receive pause frame after current transmitting frame complete. MAC automatically transmits pause frame or back pressure feature depending on fill level of RxFIFO in Full-duplex mode or in Half-duplex mode.
- Automatic transmission of pause frame on assertion and de-assertion of flow control input frame. Zero-quanta pause time length frame for Full-duplex operation. IEEE 802.3x flow control for Full-duplex operation support. Back pressure feature to the MAC core based on RxFIFO fill level (Cut-Through mode) support. IEEE 802.3x flow control for Half-duplex operation support.
- Software configurable for automatic PAD / CRC generation in transmits operation.
- Software configurable for automatic PAD / CRC stripping in receives operation.
- Software configurable for frame length.
- Software configurable for inter-frame gap.
- Support different receiving filter mode.
- IEEE 802.1Q VLAN tag detection function support for reception frames.
- Support mandatory network statistics standard (RFC2819 / RFC2665).
- Two types of wakeup frame detection: LAN remote wakeup frame and AMD Magic Packet™ frames.
- Support checking checksum (IPv4 header, TCP, UDP or ICMP encapsulated in IPv4 or IPv6 data format).
- Support Ethernet frame time stamping for both transmit and receive operation, which describes in IEEE 1588-2008, and 64 bits time stamps are given in each frame’s status.
- Two independent FIFO for transmitting and receiving.

- Support special condition frame discards handling, e.g. late collision, excessive collisions, excessive deferral or underrun.
- In the process of frame transmission, support computation and insertion of hardware checksum under store-and-forward mode.

**DMA Feature**

- Two types of descriptor addressing: Ring and Chain.
- Descriptor of transmit and receive both can transfer data up to 8192 bytes.
- Software configurable normal and abnormal interrupt for many status conditions.
- Support round-robin or fixed priority to arbitrate the request of transmit and receive controller.

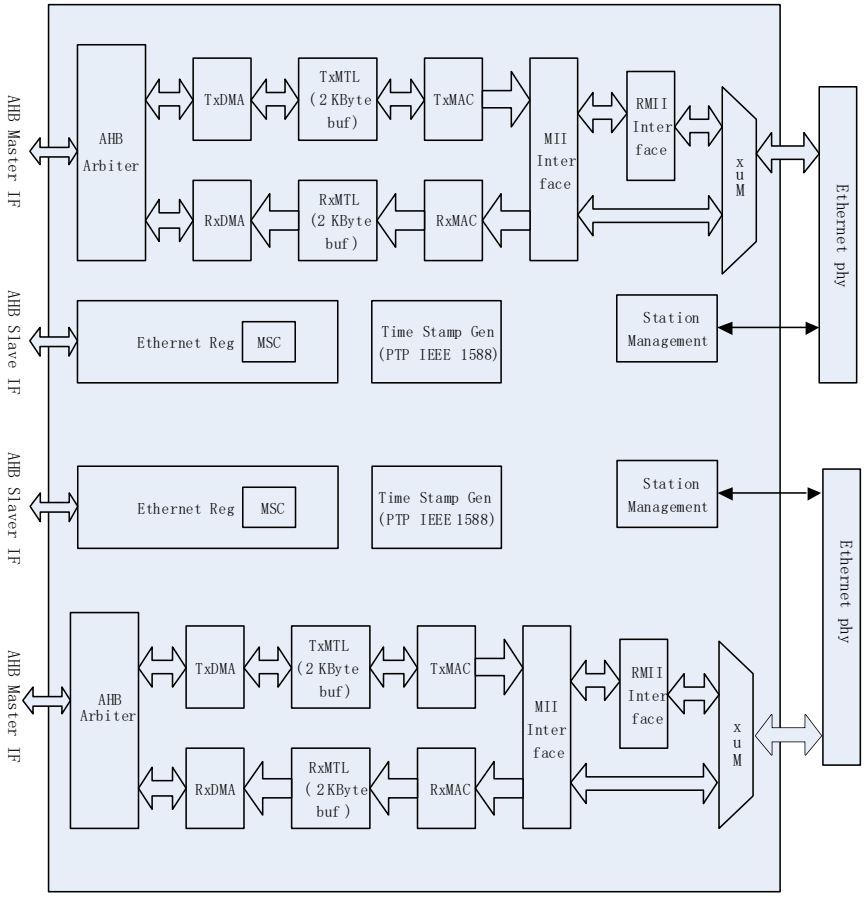
**PTP Feature**

- Support IEEE 1588 time synchronization function.
- Support two correction methods: Coarse or Fine.
- Support output pulse in seconds.
- Preset expected time reaching trigger and interrupt

### 43.2.1. Block diagram

The Ethernet module is composed of two MAC modules, two MII/RMII modules and two DMA modules by descriptor control. When using Ethernet, the user should ensure that the configured AHB clock frequency is no less than 25MHz.

**Figure 43-1. ENET module block diagram**




The MAC module is connected to the external PHY by MII or RMII through one selection bit (refer to SYSCFG_PMCFG register). The SMI (Station Management Interface) is used to configure and manage external PHY.

Transmitting data module includes:

- TxDMA controller, used to read descriptors and data from memory and writes status to memory.
- TxMTL, used to control, management and store the transmit data. TxFIFO is implemented in this module and used to cache transmitting data from memory for MAC transmission.
- The MAC transmission relative control registers, used to control frame transmit.

Receiving data module includes:

- RxDMA controller, used to read descriptors from memory and writes received frame data and status to memory.
- RxMTL, used to control, management and store reception data. RxFIFO is implemented in this module and used to temporarily store received frame data before forwarding them into the system physical memory.
- The MAC reception relative control registers, used to control frame receive and marked the receiving state. Also a receiving filter with a variety of filtering mode is implemented in MAC, used to filter out specific Ethernet frame.

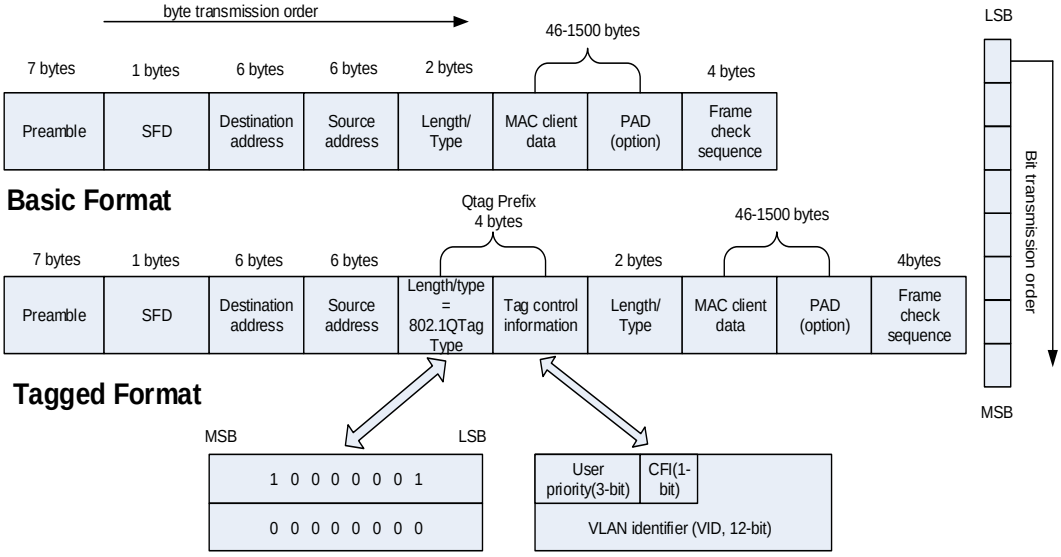
### 43.2.2. MAC 802.3 Ethernet packet description

Data communication of MAC can use two frame formats:

- Basic frame format.
- Tagged frame format.

*Figure 43-2. MAC / Tagged MAC frame format* describes the structure of the frame (Basic and Tagged) that includes the following fields:

**Figure 43-2. MAC / Tagged MAC frame format**




**Note:** The Ethernet controller transmits each byte at LSB first except FCS field.

CRC calculation data comes from all bytes in the frame except the Preamble and SFD domain. The Ethernet frame's 32-bit CRC calculation value generating polynomial is fixed 0x04C11DB7 and this polynomial is used in all 32-bit CRC calculation places in Ethernet module, as follows:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

### 43.2.3. Ethernet signal description

*Table 43-1. Ethernet signals (MII)* and *Table 43-2. Ethernet signals (RMII)* shows the MAC module that pin is used default and remapping functions and specific configuration in MII / RMII mode.

**Table 43-1. Ethernet signals (MII)**

| Signals | Pin1 | Pin2 | Pin3 | Pin mode |
|---|---|---|---|---|
| ETH0_MDC | PC1 | | | AF output push-pull(AF11) |
| ETH0_MII_TXD2 | PC2 | | | AF output push-pull(AF11) |
| ETH0_MII_TX_CLK | PC3 | | | AF output push-pull(AF11) |
| ETH0_MII_CRS | PA0 | PH2 | | AF output push-pull(AF11) |

| Signals | Pin1 | Pin2 | Pin3 | Pin mode |
|---|---|---|---|---|
| ETH0_MII_RX_CLK | PA1 | | | AF output push-pull(AF11) |
| ETH0_MDIO | PA2 | | | AF output push-pull(AF11) |
| ETH0_MII_COL | PA3 | PH3 | | AF output push-pull(AF11) |
| ETH0_MII_RX_DV | PA7 | | | AF output push-pull(AF11) |
| ETH0_MII_RXD0 | PC4 | | | AF output push-pull(AF11) |
| ETH0_MII_RXD1 | PC5 | | | AF output push-pull(AF11) |
| ETH0_MII_RXD2 | PB0 | PH6 | | AF output push-pull(AF11) |
| ETH0_MII_RXD3 | PB1 | PH7 | | AF output push-pull(AF11) |
| ETH0_PPS_OUT | PB5 | PG8 | | AF output push-pull(AF11) |
| ETH0_MII_TXD3 | PB8 | PE2 | | AF output push-pull(AF11) |
| ETH0_MII_RX_ER | PB10 | | | AF output push-pull(AF11) |
| ETH0_MII_TX_EN | PB11 | PG11 | | AF output push-pull(AF11) |
| ETH0_MII_TXD0 | PB12 | PG13 | | AF output push-pull(AF11) |
| ETH0_MII_TXD1 | PB13 | PG12 | PG14 | AF output push-pull(AF11) |
| ETH1_MDC | PG6 | | | AF output push-pull(AF6) |
| ETH1_MII_TXD2 | PG12 | | | AF output push-pull(AF6) |
| ETH1_MII_TX_CLK | PG9 | | | AF output push-pull(AF6) |
| ETH1_MII_CRS | PH15 | | | AF output push-pull(AF6) |
| ETH1_MII_RX_CLK | PH12 | | | AF output push-pull(AF6) |
| ETH1_MDIO | PH14 | | | AF output push-pull(AF6) |
| ETH1_MII_COL | PH13 | | | AF output push-pull(AF6) |
| ETH1_MII_RX_DV | PH11 | | | AF output push-pull(AF6) |
| ETH1_MII_RXD0 | PH8 | | | AF output push-pull(AF6) |
| ETH1_MII_RXD1 | PH9 | | | AF output push-pull(AF6) |
| ETH1_MII_RXD2 | PH6 | | | AF output push-pull(AF6) |
| ETH1_MII_RXD3 | PH7 | | | AF output push-pull(AF6) |
| ETH1_PPS_OUT | PG8 | | | AF output push-pull(AF6) |
| ETH1_MII_TXD3 | PG15 | | | AF output push-pull(AF6) |
| ETH1_MII_RX_ER | PH10 | | | AF output push-pull(AF6) |
| ETH1_MII_TX_EN | PG11 | | | AF output push-pull(AF6) |
| ETH1_MII_TXD0 | PG13 | | | AF output push-pull(AF6) |
| ETH1_MII_TXD1 | PG14 | | | AF output push-pull(AF6) |

**Table 43-2. Ethernet signals (RMII)**

| Signals | Pin1 | Pin2 | Pin3 | Pin mode |
|---|---|---|---|---|
| ETH0_MDC | PC1 | | | AF output push-pull(AF11) |
| ETH0_RMII_REF_CLK | PA1 | | | AF output push-pull(AF11) |
| ETH0_MDIO | PA2 | | | AF output push-pull(AF11) |
| ETH0_RMII_CRS_DV | PA7 | | | AF output push-pull(AF11) |
| ETH0_RMII_RXD0 | PC4 | | | AF output push-pull(AF11) |
| ETH0_RMII_RXD1 | PC5 | | | AF output push-pull(AF11) |
| ETH0_PPS_OUT | PB5 | | | AF output push-pull(AF11) |

| Signals | Pin1 | Pin2 | Pin3 | Pin mode |
|---|---|---|---|---|
| ETH0_RMII_TX_EN | PB11 | PG11 | | AF output push-pull(AF11) |
| ETH0_RMII_TXD0 | PB12 | PG13 | | AF output push-pull(AF11) |
| ETH0_RMII_TXD1 | PB13 | PG12 | PG14 | AF output push-pull(AF11) |
| ETH1_MDC | PG6 | | | AF output push-pull(AF6) |
| ETH1_RMII_REF_CLK | PH12 | | | AF output push-pull(AF6) |
| ETH1_MDIO | PH14 | | | AF output push-pull(AF6) |
| ETH1_RMII_CRS_DV | PH11 | | | AF output push-pull(AF6) |
| ETH1_RMII_RXD0 | PH8 | | | AF output push-pull(AF6) |
| ETH1_RMII_RXD1 | PH9 | | | AF output push-pull(AF6) |
| ETH1_PPS_OUT | PG8 | | | AF output push-pull(AF6) |
| ETH1_RMII_TX_EN | PG11 | | | AF output push-pull(AF6) |
| ETH1_RMII_TXD0 | PG13 | | | AF output push-pull(AF6) |
| ETH1_RMII_TXD1 | PG14 | PG15 | | AF output push-pull(AF6) |

## 43.3. Function overview

### 43.3.1. Interface configuration

The Ethernet block can transmit and receive Ethernet packets from an off-chip Ethernet PHY connected through the MII / RMII interface. MII or RMII mode is selected by software and carry on the PHY management through the SMI interface.

#### SMI: Station management interface

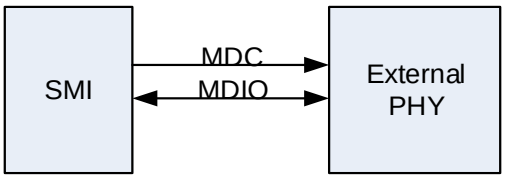
SMI is designed to access and configure PHY's configuration.

Station management interface (SMI) is performed through two wires to communicate with the external PHY: one clock line (MDC) and one data line (MDIO) . The maximum number of PHYs supported by this interface is 32. But at the same time only one register of a PHY can be accessed.

MDC and MDIO specific functions as follows:

- **MDC**: A clock of maximum frequency is 2.5 MHz. The pin remains low level when it is in idle state. The minimum high or low level lasts time of MDC must be 160ns, and the minimum period of MDC must be 400ns when it is in data transmission state.
- **MDIO**: Used to transfer data in conjunction with the MDC clock line, receiving data from external PHY or sending data to external PHY.

**Figure 43-3. Station management interface signals**



**write operation**

Applications need to write transmission data to the ENET_MAC_PHY_DATA register and operate the ENET_MAC_PHY_CTL register as follows:

1. Set the PHY device address and PHY register address, and set PW to 1, so that can select write mode;
2. Set PB bit to start transmission. In the process of transaction PB is always high until the transfer is complete. Hardware will clear PB bit automatically.

The application can be aware of whether a transaction is complete or not through checking PB bit. When PB is 1, it means the application should not change the PHY address register contents and the PHY data register contents because of operation is running. Before writing PB bit to 1, application must poll the PB bit until it is 0.

**read operation**

Applications need to operate the ENET_MAC_PHY_CTL register as follows:

1. Set the PHY device address and PHY register address and set PW to 0, so that can select read mode;
2. Set PB bit to start reception. In the process of transaction PB is always high until the transfer is complete. Hardware will clear PB bit automatically.

The application can be aware of whether a transaction is complete or not through checking PB bit. When PB is 1, it means the application should not change the PHY address register contents and the PHY data register contents because of operation is running. Before writing PB bit to 1, application must poll the PB bit until it is 0.

**Note:** Because the PHY register address 16-31 register function is defined by each manufacturer, access different PHY device's this part should see according to the manufacturer's manual to adjust the parameters of applications. Details of catalog that firmware library currently supports the PHY device can refer to firmware library related instructions.

**clock configuration**

The SMI clock is generated by dividing application clock (AHB clock). In order to guarantee the MDC clock frequency is no more than 2.5MHz, application should set appropriate division factor according to the different AHB clock frequency. ***Table 43-3. Clock range*** lists the frequency factor corresponding AHB clock selection.

**Table 43-3. Clock range**

| AHB clock | MDC clock | Bits CLR[2:0] in ENET_MAC_PHY_CTL |
|---|---|---|
| 300~350MHz | AHB clock/142 | 0x6 |
| 250~300MHz | AHB clock/124 | 0x5 |
| 150~250MHz | AHB clock/102 | 0x4 |
| 35~60MHz | AHB clock/26 | 0x3 |

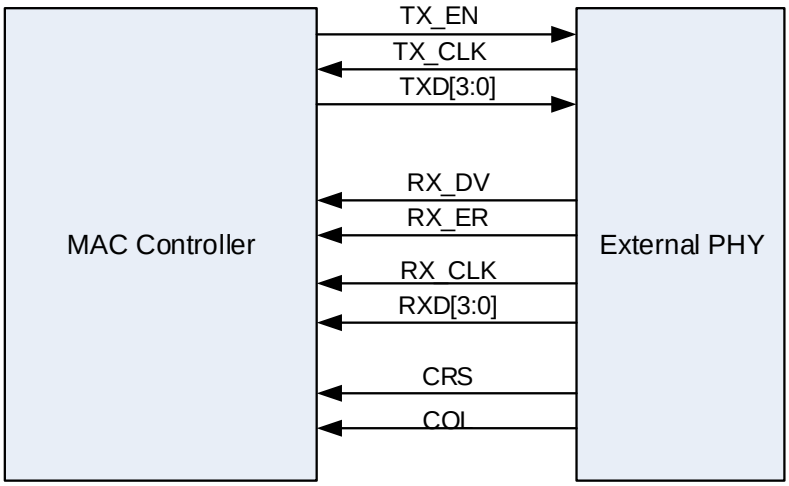
| AHB clock | MDC clock | Bits CLR[2:0] in ENET_MAC_PHY_CTL |
|---|---|---|
| 20~35MHz | AHB clock/16 | 0x2 |
| 100~150MHz | AHB clock/62 | 0x1 |
| 60~100MHz | AHB clock/42 | 0x0 |

### MII / RMII selections

Before enable the Ethernet controller clocks or when the Ethernet controller is under the reset state, the application can select the MII or RMII mode by configuring bit 23 or bit 22 of the SYSCFG_PMCFG register ENET0_PHY_SEL/ENET1_PHY_SEL. The MII mode is set by default.

### MII: Media independent interface

**Figure 43-4. Media independent interface signals**




- **MII_TX_CLK**: clock signal for transmitting data. For the data transmission speed of 10Mbit / s, the clock is 2.5MHz, for the data transmission speed of 100Mbit / s, the clock is 25MHz.

- **MII_RX_CLK**: Clock signal for receiving data. For the data transmission speed of 10Mbit / s, the clock is 2.5MHz, for the data transmission speed of 100Mbit / s, the clock is 25MHz.

- **MII_TX_EN**: Transmission enable signal. This signal must be active when the first bit of the data preamble occurs. And it needs to remain active before the all bits transmission is completed.

- **MII_TXD[3:0]**: Transmit data line, each 4 bit data transfer, data is valid when the MII_TX_EN signal is effective. The PHY would ignore the transmitted data when the MII_TX_EN signal is non-effective.

- **MII_CRS**: Carrier sense signal, only working in Half-duplex mode and controlled by the PHY. This signal does not need to be synchronized with the MII_TX_CLK and MII_RX_CLK. When it is active, means that the transmit or receive medium is not in idle state. MII_CRS signal remains active until the transmit and receive medium are both in idle state.

- **MII_COL**: Collision detection signal, only working in Half-duplex mode, controlled by the PHY. This signal does not need to be synchronized with the MII_TX_CLK and MII_RX_CLK. It is active when a collision on the medium is detected and it will remain active while the collision condition continues.

- **MII_RXD[3:0]**: Receive data line, each 4 bit data transfer; data are valid when the MII_RX_DV signal is effective. Depending on the state of MII_RX_DV and MII_RX_ER, the MII_RXD[3:0] value can be used to convey some specific information (see ***Table 43-4. Rx interface signal encoding***).

- **MII_RX_DV**: Receive data valid signal, controlled by the PHY. This signal must be active when the first 4-bits of the frame data occurs. And it needs to remain active before the all bits transmission is completed. It must be inactive prior to the first clock cycle that follows the final 4-bit. MII_RX_DV signals should be effective before the SFD field appearing to ensure that receive the correct frame.

- **MII_RX_ER**: Receive error signal. In order to indicate that MAC detected an error in the receiving process, the MII_RX_ER signal must remain effective for one or more clock cycles (MII_RX_CLK). The specific error reason needs to cooperate with the state of the MII_RX_DV and the MII_RXD[3:0] data value (see ***Table 43-4. Rx interface signal encoding***).

**Table 43-4. Rx interface signal encoding**

| Signal | Normal inter-frame | | Normal reception frame data | False carrier indication | Data reception with errors |
|---|---|---|---|---|---|
| MII_RX_ER | 0 | 1 | 0 | 1 | 1 |
| MII_RX_DV | 0 | 0 | 1 | 0 | 1 |
| MII_RXD[3:0] | 0000 to 1111 | 0000 | 0000 to 1111 | 1110 | 0000 to 1111 |

#### MII clock sources

The user needs to provide an external 25MHz clock to the external PHY to generate both TX_CLK and RX_CLK clock. This 25MHz clock does not require the same one with MAC clock. It can use the external 25MHz crystal or the output clock of microcontroller's CK_OUT0 pin. If the clock source is selected from CK_OUT0 pin, the MCU needs to configure the appropriate PLL to ensure the output frequency of CK_OUT0 pin is 25MHz.

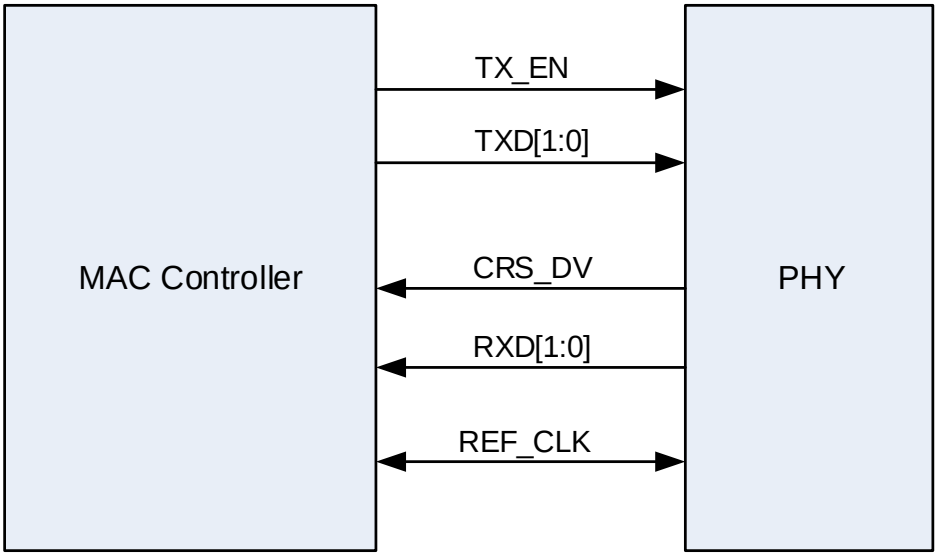
### RMII: Reduced media independent interface

The reduced media-independent interface (RMII) specification reduces the pin count during Ethernet communication. The MII specification defines 16 pins for data and control, according to the IEEE 802.3 standard. The RMII specification is dedicated to reduce the pin count to 7.

RMII characteristics:

- The clock signal needs to be increased to 50MHz and only one clock signal.
- MAC and external PHY use the same clock source.
- Using the 2-bit wide data transceiver.

**Figure 43-5. Reduced media-independent interface signals**




**RMII clock sources**

To ensure the synchronization of the clock source, the same clock source is selected for the MAC and external PHY which is called REF_CLK. The REF_CLK input clock can be connected to the external 50MHz crystal or microcontroller CK_OUT0 pin. If the clock source is from CK_OUT0 pin, then the MCU needs to configure the appropriate PLL to ensure the output frequency of CK_OUT0 pin is 50MHz.

### MII / RMII bit transmission order

No matter which interface (MII or RMII) is selected, the bit order of transmit / receive is LSB first.

The deference between MII and RMII is bit number and sending times. MII is low 4bits first and then high 4bits, but RMII is the lowest 2bits, low 2bits, high 2bits and the highest 2bits.

For example: a byte value is: 10011101b (left to right order: high to low).
Transmission order for MII use 2 cycles: 1101 -> 1001 (left to right order: high to low, 1101 corresponding to MII_T/RXD[3] to MII_T/RXD[0]).
Transmission order for RMII use 4 cycles: 01 -> 11 -> 01 -> 10 (left to right order: high to low, 01 corresponding to RMII_T/RXD[1] to RMII_T/RXD[0]).

## 43.3.2. MAC function overview

The MAC module can work in two modes (Half-duplex mode and Full-duplex mode). The Half-duplex mode, with the CSMA/CD algorithm to contend for using of the physical medium, at the same time only one transmission direction is active between two stations is active. The Full-duplex mode, simultaneous transmission and reception without any conflict mode, if all of the following conditions are satisfied: 1) PHY supports the feature of transmission and reception operations at the same time. 2) Only two devices connect to the LAN and the two devices are both configured for Full-duplex mode.

MAC module can achieve the follows functions: 1) The data packaging (transmission and reception), that includes detecting / decoding frame and delimitating frame boundary; handling source address and destination address; detecting error conditions. 2) The Medium access management in Half-duplex mode, that includes allocating medium in order to prevent conflicts; deal with conflicts.

### Transmission process of MAC

All transactions are controlled by the dedicated DMA controller and MAC in Ethernet. After received the sending instruction, the TxDMA fetches the transmit frames from system memory and pushes them into the TxFIFO, then the data in TxFIFO are poped to MAC for sending on MII / RMII interface. The method of popping is according to the selected TxFIFO mode (Cut-Through mode or Store-and-Forward mode, the specific definition sees the next paragraph). For convenient, application can configure automatically hardware calculated CRC and insert it to the FCS domain of Ethernet frame function. The entire transmission process complete when the MAC received the frame termination signal from transmit FIFO. When transmission completed, the transmission status information will be composed of MAC and write return to the DMA controller, the application can query through the DMA current transmit descriptor.

Operation for popping data from FIFO to the MAC has two modes:

- In Cut-Through mode, the data in FIFO is ready to be popped to MAC once the number of bytes in the FIFO exceeds or equals the configured threshold level or when the end-of-frame flag in descriptor is written. User can configure the threshold level through the TTHC[2:0] in ENET_DMA_CTL
- In Store-and-forward mode, the data is ready to be popped to the MAC core only after complete frame is stored in the FIFO. But there is another condition where the frame is not completely written into the FIFO, and FIFO will also take out data. This is when the transmitted Ethernet frame is bigger than FIFO size, the frame is popped towards the MAC before the transmit FIFO becomes full.

#### Handle special cases

In the transmission process, due to the insufficient TxDMA descriptor or misuse of FTF bit in ENET_DMA_CTL register (when this bit is set, it will clear FIFO data and reset the FIFO pointer, after clear operation is completed, it will be reset), there will be a transmit data underflow fault occurs because of insufficient data in FIFO. At the same time MAC will identify such data underflow state and write relevant status flag.

If one transmit frame uses two TxDMA descriptors for sending data, then the first segment (FSG) and the last segment (LSG) of the first descriptor should be 10b and the second ones should be 01b.If both the FSG bit of the first and the second descriptor are set and the LSG bit in the first descriptor is reset, then the FSB bit of the second descriptor will be ignored and these two descriptors are considered to sending the only one frame.

If the byte length of one transmission MAC frame’s data field is less than 46 (for Tagged MAC frame is less than 42), application can configure the MAC for automatically adding a load of content of ‘0’ bit to make the byte length of frame’s data field in accordance with the relevant

domain of definition of IEEE802.3 specification. At the same time, if automatically adding zeros function is performed, the MAC will certainly calculate CRC value of the frame and append it to the frame's FCS field domain no matter what configuration of DCRC bit in the descriptor is.

## Transmission management of MAC

### Jabber timer

In case of one station occupies the PHY for a long time, there is a jabber timer designed for cutting off the frame whose length is more than 2048 bytes. By default, jabber timer is enabled so when application is transmitting a frames whose byte length is more then 2048, the MAC will only transmit 2048 bytes and drop the last ones.

### Collision condition solve mechanism – Re-transmission

When the MAC is running under Half-duplex mode, collision may happen when MAC is transmitting frame data on interface. When no more than 96 bytes data popped from FIFO towards MAC and collision condition occurs, the re-transmission function is active. In this case, MAC will stop current transmitting and then read frame data from FIFO again and send them on interface again. When more than 96 bytes data popped from FIFO towards MAC and collision condition occurs, MAC will abort transmitting current frame data and not re-transmit it. Also MAC will set late collision flag in descriptor to inform application.

### Transmit FIFO flush operation

Application can clear TxFIFO and reset the FIFO data pointer by setting FTF bit of ENET_DMA_CTL register. The flush operation will be executed at once no matter whether TxFIFO is popping data to MAC. This results in an underflow event in the MAC transmitter, and the makes frame transmission abort. At the same time, MAC returns state information of frame and transmit status words transferred to the application. The status of such a frame is marked with both underflow and frame flush events (FRMF and UFE bits in Transmit Descriptor0). When the transmit data in TxFIFO is flushed, the transmit status word will be written back to descriptor. After the status is written, the flush operation is complete. When a flush operation is received, all the following data which should be popped from TxFIFO into MAC will be dropped unless a new FSG bit of descriptor is received. After operation completed, the FTF bit of ENET_DMA_CTL register is then automatically cleared.

### Transmit inter-frame gap management

MAC can manage the interval time between two frames. This interval time is called frame gap time. For Full-duplex mode, after complete sending a frame or MAC entered idle state, the gap time counter starts counting. If another transmit frame presents before this counter has not reach the configured IGBS bit time in ENET_MAC_CFG register, this transmit frame will be pended unless the counter reach the gap time. But if the second transmit frame presents after the gap time counter has reached the configured gap time, this frame will send immediately. For Half-duplex mode, the gap time counter follows the Truncated Binary

Exponential Backoff algorithm. Briefly speaking, the gap time counter starts after the previous frame has completed transmitting on interface or the MAC entered idle state, and there are three conditions may occur during the gap time:

- The carrier sense signal active in the first 2 / 3 gap period. In this case, the counter will reload and restart.
- The carrier sense signal active in the last 1 / 3 gap period. In this case, the counter will not reload but continue counting, and when reaches gap time, the MAC sends the second frame.
- The carrier sense signal not active during the whole gap period. In this case, the counter stops after reaches the configured gap time and sends frame if the second frame has pended.

### Address receive module

The MAC filter is divided into error filtering (such as too short frame, CRC error and other bad frame filtering) and address filtering. This section mainly describes the address filtering.Address filtering use the static physical address (MAC address) filter and hash list filter for implementing the function. If the FAR bit in the ENET_MAC_FRMF register is '0' (by default), only the frame passed the filter will be received. This function is configured according to the parameters of the application (frame filter register) to filter the destination or / and source address of unicast or multicast frame (The difference between an individual address and a group address is determined by I / G bit in the destination address field) and report the result of the corresponding address filtering. The frame not pass through the filter will be discarded.

**Note:** If the FAR bit in the ENET_MAC_FRMF register is set to 1, frames are all thought passed the filter. In this case, even the filter result will also be updated in receive descriptor but the result will not affect whether current frame passes the filter or not.

#### Unicast frame destination address filter

For a unicast frame, application has two modes for filtering: the one is using static physical address (by setting HUF bit to ‘0’), the other is using hash list (by setting HUF bit to ‘1’).

- Static physical address (SPA) filtering.

In the filter mode, MAC supports using four MAC addresses for unicast frame filtering. In this way, the MAC compares all 6 bytes of the received unicast address to the programmed MAC address. MAC address 0 is always used and MAC address 1 to address 3 can be configured to use or not. Each byte of MAC address 1 to MAC address 3 register can be masked for comparison with the corresponding destination address byte of received frame by setting the corresponding mask byte bits (MB) in the corresponding register.

- Hash list filtering

In this filter mode, MAC uses a HASH mechanism. MAC uses a 64-bit hash list to filter the received unicast frame. This filter mode obeys the followings two filtering steps:

1. The MAC calculates the CRC value of the received frame's destination address.
2. Using the high 6 bits of the calculated CRC value as the index to retrieve the hash list. If the corresponding value of hash list is 1, the received frame passes through the filter, conversely, fail the filter.

The advantage of this type of filter is that it can cover any possible address just using a small table. But the disadvantage is that the filter is imperfect and sometimes the frames should be dropped are also be received by mistake.

**Unicast frame source address filter**

Enable MAC address 1 to MAC address 3 register and set the corresponding SAF bit in the MAC address high register, the MAC compares and filter the source address (SA) field in the received frame with the values programmed in the SA registers. MAC also supports the group filter on the source address. If the SAFLT bit in frame filter register ENET_MAC_FRMF is set, MAC drops the frame that failed the source address filtering; meanwhile the filter result will reflect by SAFF bit in Receive Descriptor0 of DMA receive descriptor. When the SAFLT bit is set, the destination address filter is also at work, so the result of the filter is simultaneous determined by DA and SA filter. This means that, as long as a frame does not pass any one of the filters (DA filter or SA filter), it will be discarded. Only a frame passing the entire filter can be forwarded to the application.

**Multicast frame destination address filter**

Application can enable the multicast frame MAC address filtering by cleaning the MFD bit in register ENET_MAC_FRMF. By configuring the value of HMF bit in ENET_MAC_FRMF register application can choose two ways just like unicast destination address filtering for address filtering.

**Broadcast frame destination address filter**

At default, the MAC unconditionally receives the broadcast frames. But when setting BFRMD bit in register ENET_MAC_FRMF, MAC discards all received broadcast frames.

**Hash or perfect address filter**

By setting the HPFLT bit in the ENET_MAC_FRMF register and setting the corresponding HUF (for unicast frame) or HMF (for multicast frame) bit in the ENET_MAC_FRMF register, the destination address (DA) filter can be configured to pass a frame when its DA matches either the hash list filter or the static physical address filter.

**Reverse filtering operation**

MAC can reverse filter-match result at the final output whether the destination address filtering or source address filtering. By setting the DAIFLT and SAIFLT bits in ENET_MAC_FRMF register, this address filter reverse function can be enabled. DAIFLT bit is used for unicast and multicast frames' DA filtering result, SAIFLT bit is used for unicast and multicast frames SA filtering result.

The ***Table 43-5. Destination address filtering table*** and ***Table 43-6. Source address filtering table*** summarize the destination address and source address filters working condition at different configuration.

**Table 43-5. Destination address filtering table**

| Frame Type | PM | HPFLT | HUF | DAIFLT | HMF | MFD | BFRMD | DA filter operation |
|---|---|---|---|---|---|---|---|---|
| Broadcast | 1 | - | - | - | - | - | - | Pass |
| | 0 | - | - | - | - | - | 0 | Pass |
| | 0 | - | - | - | - | - | 1 | Fail |
| Unicast | 1 | - | - | - | - | - | - | Pass all frames |
| | 0 | - | 0 | 0 | - | - | - | Pass on perfect / group filter match |
| | 0 | - | 0 | 1 | - | - | - | Fail on perfect / group filter match |
| | 0 | 0 | 1 | 0 | - | - | - | Pass on hash filter match |
| | 0 | 0 | 1 | 1 | - | - | - | Fail on hash filter match |
| | 0 | 1 | 1 | 0 | - | - | - | Pass on hash or perfect / group filter match |
| | 0 | 1 | 1 | 1 | - | - | - | Fail on hash or perfect / group filter match |
| Multicast | 1 | - | - | - | - | - | - | Pass all frames |
| | - | - | - | - | - | 1 | - | Pass all frames |
| | 0 | - | - | 0 | 0 | 0 | - | Pass on perfect / group filter match and drop PAUSE control frames if PCFRM = 0x |
| | 0 | 0 | - | 0 | 1 | 0 | - | Pass on hash filter match and drop PAUSE control frames if PCFRM = 0x |
| | 0 | 1 | - | 0 | 1 | 0 | - | Pass on hash or perfect / group filter match and drop PAUSE control frames if PCFRM = 0x |
| | 0 | - | - | 1 | 0 | 0 | - | Fail on perfect / group filter match and drop PAUSE control frames if PCFRM = 0x |
| | 0 | 0 | - | 1 | 1 | 0 | - | Fail on hash filter match and drop PAUSE control frames if PCFRM = 0x |
| | 0 | 1 | - | 1 | 1 | 0 | - | Fail on hash or perfect / group filter match and drop PAUSE control frames if PCFRM = 0x |

**Table 43-6. Source address filtering table**

| Frame type | PM | SAIFLT | SAFLT | SA filter operation |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Unicast | 1 | - | - | Pass all frames |
| | 0 | 0 | 0 | Pass status on perfect / group filter match but do not drop frames that fail |
| | 0 | 1 | 0 | Fail status on perfect / group filter match but do not drop frame |
| | 0 | 0 | 1 | Pass on perfect / group filter match and drop frames that fail |
| | 0 | 1 | 1 | Fail on perfect / group filter match and drop frames that fail |

**Promiscuous mode**

If the PM bit in ENET_MAC_FRMF register is set, promiscuous mode is enable. Then the address filter function is bypassed, all frames are thought passed through the filter. At the same time the receive status information DA / SA error bit is always '0'.

**Pause control frame filter**

When MAC received pause frame, it will detect 6 bytes DA field in the frame. If UPFDT bit in ENET_MAC_FCTL register is 0, it is determined by whether the value of the DA field conforms to the unique value (0x0180C2000001) with IEEE-802.3 specification control frames. If UPFDT bit in ENET_MAC_FCTL register is set, MAC additionally compares DA field with the programmed MAC address for bit match. If DA field match and receive flow control is enabled (RFCEN bit in ENET_MAC_FCTL register is set), the corresponding pause control frame function will be triggered. Whether this filter passed pause frame is forwarded to memory is depending on the PCFRM[1:0] bit in ENET_MAC_FRMF register.

### Reception process of MAC

Received frames will be pushed to the RxFIFO. The MAC strips the preamble and SFD of the frame, and starts pushing the frame data beginning with the first byte following the SFD to the RxFIFO. If IEEE 1588 time stamp function is enabled, the MAC will record the current system time when a frame's SFD is detected. If the frame passes the address filter, this time stamp is passed on to the application by writing it to descriptor.

The MAC can automatically strip PAD and FCS field data when the length / type field of received frame is less than 1536 if APCD bit is set. MAC pushes the data of the frame into RxFIFO up to the count specified in the length / type field, then starts dropping bytes (including the FCS field). If the value of length / type field is greater than or equal to 0x600, the automatically strip FCS field function is configured by the TFCD bit regardless of APCD.

If the watchdog timer is enabled (WDD bit in ENET_MAC_CFG is reset), a frame has more than 2048 bytes will be cut off receiving when has received 2048 bytes. If the watchdog timer is disabled, the MAC can extend the max receiving data bytes to 16384, any data beyond this number will be cut off.

When RxFIFO works at Cut-Through mode, it starts popping out data from RxFIFO when the number of FIFO is greater than threshold value (RTHC bits in ENET_DMA_CTL register).

After all data of a frame pop out, receive status word is sent to DMA for writing back to descriptor. In this mode, if a frame has started to forward to application by DMA from FIFO, the forwarding will continue until the frame is end even if frame error is detected. Although the error frame is not discarded, the error status will reflect in descriptor status field.

When RxFIFO works at Store-and-Forward mode (set by RSFD bit in ENET_DMA_CTL), DMA reads frame data from RxFIFO only after RxFIFO has completed received the whole frame. In this mode, if the MAC is configured to discard all error frames, then only valid frames without any error can be read out from RxFIFO and forward to the application. Once the MAC detects an SFD signal on the interface, a receive operation is started. The MAC strips the preamble and SFD before processing the frame. The header fields are checked by filtering and the FCS field used to verify the CRC for the frame. The frame is discarded by MAC if it fails to pass the address filter.

## Reception management of MAC

### Receive operation on multi-frame handling

It is different from transmit operation, after receiving the last byte of a frame, the MAC can judge the status of the receiving operation, so the second received frame's forwarding is surely followed by the first received frame data and status.

### Error handling

- If RxFIFO becomes full but the last received byte is not the end of frame (EOF), the RxFIFO will discard the whole frame data and return an overflow status. Also the counter of counting the overflow condition times will plus 1.
- If the RxFIFO is configured in Store-and-Forward mode, the MAC can filter and discard all error frames. But according to the configuration of FERF and FUF bit in ENET_DMA_CTL register, RxFIFO can also receive and forward such error frame and the frame that length is less than the minimum length.
- If the RxFIFO is configured in Cut-Through mode, not all the error frames can be dropped. Only when the start of frame (SOF) has not been read from RxFIFO and the receive frame has been detected error status, the RxFIFO will discard the whole error frame.

## Flow control module

The MAC manages transmission frame through back pressure (in Half-duplex mode) and the pause control frame (in Full-duplex mode) for flow control.

- Half-duplex mode flow control: Back Pressure

When MAC is configured in Half-duplex mode, there are two conditions to trigger the back pressure feature. Both of the two conditions are triggered to enable back pressure function which is implemented by sending a special pattern (called jam pattern) 0x5555 5555 once to notify conflict to all other sites. The first condition is triggered by application setting the FLCB / BKPA bit in ENET_MAC_FCTL register. The second condition occurs during receiving frame. When MAC receiver is receiving frame, the byte number of RxFIFO is more and more great.

When this number is greater than the high threshold (RFA bits in ENET_MAC_FCTH), MAC will set the back pressure pending flag. If this flag is set and a new frame presents on interface, MAC will send a jam pattern to delay receiving this new frame a back pressure time. After this back pressure time is end, external PHY will send this new frame again. If the number of the RxFIFO is not less than low threshold (RFD bits in ENET_MAC_FCTH) during this back pressure time, a jam pattern is send again. If the number of the RxFIFO is less than low threshold (RFD bits in ENET_MAC_FCTH) during this back pressure time, MAC resets the back pressure pending flag and is enable to receive the new frame instead of sending jam pattern.

- Full-duplex mode flow control: Pause Frame

The MAC uses a mechanism named "pause frame" for flow control in Full-duplex mode. Receiver can send a command to the sender for informing it to suspend transmission a period of time. If the application sets transmit flow control bit TFCEN in ENET_MAC_FCTL register, MAC will generate and transmit a pause frame when either of two conditions is satisfied in Full-duplex mode. There are two conditions to start transmit pause frames:

1. Application sets FLCB / BKPA bit in ENET_MAC_FCTL register to immediately send a pause frame. When doing this, MAC sends a pause frame right now with the pause time value PTM configured in ENET_MAC_FCTL register. If application considers the pause time is no need any more because the transmit frame can be transmitted without pause time, it can end the pause time by setting the pause time value PTM bits in ENET_MAC_FCTL register to 0 and set FLCB / BKPA bit to send this zero pause time frame.

2. MAC automatically sends pause time when the RxFIFO is in some condition. When MAC is receiving frame, RxFIFO will be fill in many receive data. At same time RxFIFO pops out data to RxDMA for forwarding to memory. If the popping frequency is lower than MAC pushing frequency, the number of bytes in RxFIFO is getting great. Once the data amount in RxFIFO is greater than the active threshold value (RFA bits in ENET_MAC_FCTH) of flow control, MAC will send a pause frame with PTM value in it. After sending pause frame, MAC will start a counter with configured reload value PLTS in ENET_MAC_FCTL register, when configured PLTS time has spent, the MAC will check RxFIFO again. If the byte number in RxFIFO is also greater than active threshold value, the MAC sends a pause time again. When the byte number of RxFIFO is lower than the de-active threshold value, MAC maybe send a pause frame with zero time value in frame’s pause time field if DZQP bit in ENET_MAC_FCTL register is reset. This zero-pause time frame can inform send station that RxFIFO is almost empty and can receive new data again.

The MAC manages reception frames through follow method for flow control:In Full-duplex mode, the MAC can detect the pause control frames, and perform it by suspending a certain time which is indicated in pause time field of detected pause control frame and then to transmit data. This function can set by RFCEN bit in ENET_MAC_FCTL register. If this function is not enabled, the MAC will ignore the received pause frames. If this function is enabled, MAC can decode this frame. Type field, opcode field and pause time field in the frame are all recognized

by the MAC. During the pause time period, if MAC received a new pause frame, the new pause time filed value is loaded to the pause time counter immediately. If the new pause time filed is zero, then the pause time counter stops and transmit operation recovers. Application can configure PCFRM bit in ENET_MAC_FRMF register to decide the solving method for such control frame.

### Checksum offload engine

The MAC supports transmit checksum offload. This feature can calculate checksum and insert it in the transmit frame, and detect error in the receive frame.

The follows describes the operation of transmit checksum offload.

**Note:** This function is enabled only when the TSFD bit in the ENET_DMA_CTL register is set (TxFIFO is configured to Store-and-Forward mode) and application must ensure the TxFIFO deep enough to store the whole transmit frame. If the depth of the TxFIFO is less than the frame length, the MAC only does calculation and insertion for IPv4 header checksum field.

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460 and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6 and ICMPv6 packet header specifications, respectively.

- IP header checksum

If the value is 0x0800 in type field of Ethernet frame and the value is 0x4 in the IP datagram's version field, checksum offload module marks the frame as IPv4 package and calculated value replace the checksum field in frame. Because of IPv6 frame header does not contain checksum field, the module will not change any value of the IPv6's header field. After IP header checksum calculation end, the result is stored in IPHE bit (In Transmit Descriptor0). The following shows the conditions under which the IPHE bit can be set:

- For IPv4 frame type:

A) . Type field is 0x0800 but version filed in IP header is not 0x4.
B) . IPv4 header length field value is greater than total frame byte length.
C) . The value of IPv4 header length field is less than 0x5 (20 bytes).

- For IPv6 frame type:

A) . Type field is 0x86dd but version field in IP header is not 0x6.
B) . Before the IPv6 standard header or extension header has been completely received the frame is end. The length of IPv6 standard header is 40 bytes, and the extension header contains corresponding header length field.

- TCP / UDP / ICMP payload checksum

The checksum offload module processes the IPv4 or IPv6 header (including extension headers) and marks the type of frame (TCP, UDP or ICMP).

But when the following frame cases are detected, the checksum offload function will be bypassed and these frames will not be processed by the checksum offload module:

- Incomplete IPv4 or IPv6 frames.

- IP frames with security features (e.g. authentication header, security payload).
- IP frames without TCP / UDP / ICMPv4 / ICMPv6 payload.
- IPv6 frames with routing headers.

The checksum offload module calculates the payload (TCP, UDP, or ICMP) and inserts the result into its corresponding field in the header. It has two modes when working, as follows:

1. The checksum calculation does not include TCP, UDP, or ICMPv6 pseudo-headers and assumes that the checksum field of the input frame already has the value. The checksum calculation includes checksum field, and the value of the original checksum field is replaced after the calculation is completed.

2. Checksum offload module clears the contents of the checksum field in the transmission frame and make calculation which includes TCP, UDP, or ICMPv6 pseudo-header data and will instead the transmission frame's original checksum field by the final calculation results.

After calculated by checksum offload module, the result can be found in IPPE bit of Transmit Descriptor0. The following shows the conditions under which the IPPE bit can be set:

1. In Store-and-Forward mode, frame has been forwarded to MAC transmitter but no EOF is written to TxFIFO.

2. Frame is ended but the byte numbers which the payload length field of the frame indicates has not been reached.

If the packet length is greater than the marked length, checksum module does not report errors, the excess data will be discarded as padding bytes. If the first condition of IPPE error is detected, the value of the checksum does not insert a TCP, UDP or ICMP header. If the second condition of IPPE error is detected, checksum calculation results will still insert the appropriate header fields.

**Note:** For ICMP packets over IPv4 frame, the checksum field in the ICMP packet must always be 0x0000 in both modes due to such packets are not defined pseudo-headers.The follows describes the operation of receive checksum offload.

Receive checksum offload is enabled when IPFCO bit in ENET_MAC_CFG register is set. Receive checksum offload can calculate the IPv4 header checksum and check whether it matches the contents of the IPv4 header checksum field. The MAC identifies IPv4 or IPv6 frames by checking for the value of 0x0800 or 0x86DD respectively in the received Ethernet frame type field. This method is also used to identify frames with VLAN tags. Header checksum error bits in DMA receive descriptor (the IPHERR bit in Receive Descriptor0) reflects the header checksum result. This bit is set if received IP header has the following errors:

- Any mismatch between the IPv4 calculation result by checksum offload module and the value in received frame's checksum field.
- Any inconsistent between the data type of Ethernet type field and IP header version field.

- Received frame length is less than the length indicated in IPv4 header length field, or IPv4 or IPv6 header is less than 20 bytes.

Receive checksum offload also identifies the data type of the IP packet is TCP, UDP or ICMP, and calculate their checksum according to TCP, UDP or ICMP specification. Calculation process can include data of TCP / UDP / ICMPv6 pseudo-header. Payload checksum error bits in DMA receive descriptor (the PCERR bit in Receive Descriptor0) reflects the payload checksum result. This bit is set if received IP payload has the following errors:

- Any mismatch between the TCP, UDP or ICMP checksum calculation result by checksum offload and the received TCP / UDP / ICMP frame's checksum field.
- Any inconsistent between the received TCP, UDP or ICMP data length and length of IP header.

The received checksum offload does not calculate the following conditions: Incomplete IP packets, IP packets with security features, packets of IPv6 routing header and data type is not TCP, UDP or ICMP.

#### MAC loopback mode

Often, loopback mode is used for testing and debugging hardware and software system for application. The MAC loopback mode is enabled by setting the LBM bit in ENET_MAC_CFG register. In this mode, the MAC transmitter sends the Ethernet frame to its own receiver. This mode is disabled by default.

### 43.3.3. DMA controller description

Ethernet DMA controller is designed for frame transmission between FIFO and system memory which can reduce the occupation of CPU. Communication between the CPU and the DMA is achieved by the two kinds of data structures. Which are descriptor table (ring or chain type) and data buffer, and control and status register.

Applications need to provide the memory for storage of descriptor tables and data buffers. Descriptors that reside in the memory act as pointers to these buffers. Transmission has transmission descriptor and reception has reception descriptor. The base address of each table is stored in ENET_DMA_TDTADDR and ENET_DMA_RDTADDR register. Descriptors of transmission constituted by 4 descriptor word (Transmit Descriptor0-3) when DFM=0 and 8 descriptor word (Transmit Descriptor0-7) when DFM=1 (Enhanced descriptor mode). Likewise, reception descriptors constituted by 4 descriptor word (Receive Descriptor0-3) when DFM=0 and 8 descriptor word (Receive Descriptor0-7) when DFM=1. Each descriptor can point to a maximum of two buffers. The value of the buffer 2 can be programmed to the second data address or the next descriptor address which is determined by the configured descriptor table type: Ring or Chain. Buffer space only contains frame data which are located in host's physical memory space. One buffer can store only one frame data but one frame data can be stored in more than one buffer which means one buffer can only store a part of a frame. When chain structure is set, descriptor table is an explicitly one and when ring structure is set, descriptor table is an implicitly one. Explicit chaining of descriptors is

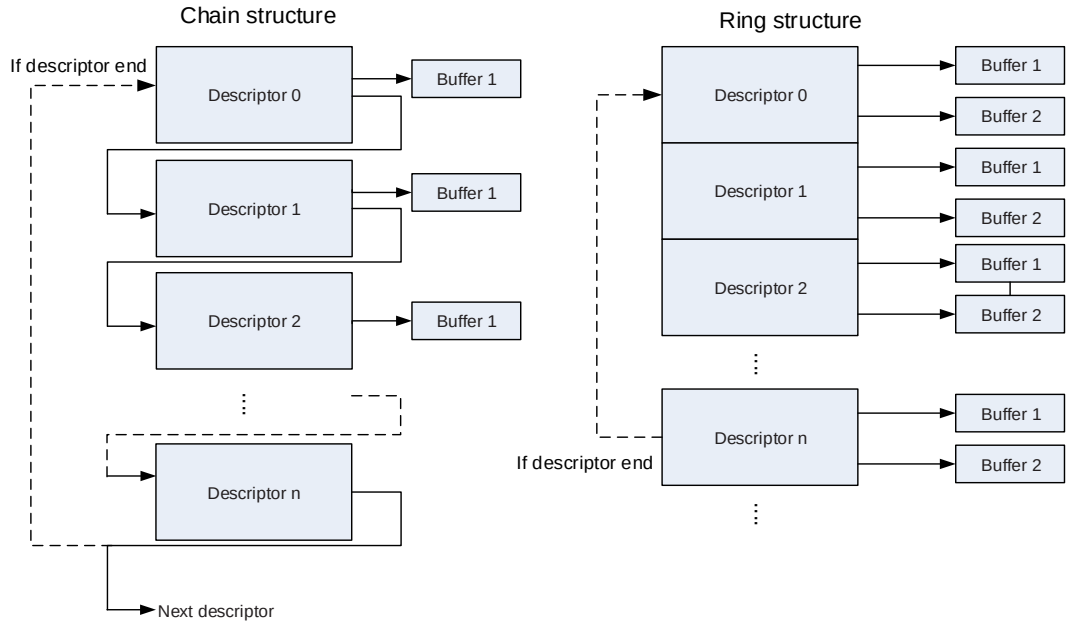
accomplished by configuring the second address chained in both receive and transmit descriptors (configure RCHM bit in the Receive Descriptor1 and TCHM bit in the Transmit Descriptor0), at this time Receive Descriptor2 and Transmit Descriptor2 are stored the data buffer address, Receive Descriptor3 and Transmit Descriptor3 should be stored the next descriptor address, this connection method of descriptor table is called chain structure. Implicitly chaining of descriptors is accomplished by clearing the RCHM bit in the Receive Descriptor1 and TCHM bit in the Transmit Descriptor0, at this time Receive Descriptor2/Transmit Descriptor2 and Receive Descriptor3/Transmit Descriptor3 should be all stored the data buffer address, this connection method of descriptor table is called ring structure. When current descriptor's buffer address is used, descriptor pointer will point to the next descriptor. If chain structure is selected, the pointer points to the value of buffer 2. If ring structure is selected, the pointer points to an address calculated as below:

DFM=0: Next descriptor address = Current descriptor address + 16 + DPSL * 4

DFM=1: Next descriptor address = Current descriptor address + 32 + DPSL * 4

If current descriptor is the last one in descriptor table, application needs to set the TERM bit in Transmit Descritor0 or RERM bit in Receive Descritor1 to inform DMA the current descriptor is the last one of the table in ring structure. At this time, the next descriptor pointer points back to the first descriptor address of the descriptor table. In chain structure, can also set Receive Descriptor3 and Transmit Descriptor3 value to point back to the first descriptor address of the descriptor table. The DMA skips to the next frame buffer when the end of frame is detected.

**Figure 43-6. Descriptor ring and chain structure**




## Alignment rule for data buffer address

The DMA controller supports all alignment types: byte alignment, half-word alignment and word alignment. This means application can configure the buffer address to any address. But during the operation of the DMA controller, access address is always word align and is

different between write and read access. Follow example describes the detail:

- Buffer Reading: Assuming the transmit buffer address is 0x2000 0AB2, and 15 bytes need to be transferred. After starting operating, the DMA controller will read five word addresses which are 0x2000 0AB0, 0x2000 0AB4, 0x2000 0AB8, 0x2000 0ABC and 0x2000 0AC0. But when sending data to the FIFO, the first two bytes (0x2000 0AB0 and 0x2000 0AB1) and the last 3 bytes (0x2000 0AC1, 0x2000 0AC2 and 0x2000 0AC3) will be dropped.
- Buffer Writing: Assuming the receive buffer address is 0x2000 0CD2, and 16 bytes need to be stored. After starting operating, the DMA controller will write five times 32-bit data from address 0x2000 0CD0 to 0x2000 0CE0. But the first 2 bytes (0x2000 0CD0 and 0x2000 0CD1) and the last 2 bytes (0x2000 0CE2 and 0x2000 0CE3) will be substituted by the virtual bytes.

**Note:** DMA controller will not write any data out of the defined buffer range.

### The effective length of the buffer

For the frame transmitting process, the effective length of the buffer is the same as the value configured by application in Transmit Descriptor1. As mentioned before, a transmitting frame can use one or more descriptors to indicate the frame information which means a frame data can be located in many buffers. When the DMA controller reads a descriptor which the FSG bit in Transmit Descriptor0 is set, it knows the current buffer is pointing to a new frame and the first byte of the frame is included. When the DMA controller reads a descriptor with FSG bit and LSG bit in Transmit Descriptor0 are both reset, it knows the current buffer is pointing to a part of current frame. When the DMA controller reads a descriptor with LSG bit in Transmit Descriptor0 is set, it knows the current buffers is pointing to the last part of the current frame. Normally one frame is stored only in one buffer (because buffer size is large enough for a normal frame), so FSG bit and LSG bit are set in the same descriptor.

For the frame receiving process, the receive buffer size must be word align. But for word-align buffer address or not word-align buffer address, the operation is different from transmitting. When the receive buffer address is word align, it's no difference with transmitting process, the effective length of the buffer is the same as the value configured by application in Receive Descriptor1. When the receive buffer address is not word align, the effective length of the buffer is less than the value configured by application in Receive Descriptor1. The effective length of the buffer should be the size value minus the low two bits value of buffer address. For example, assuming the total buffer size is 2048 bytes and buffer address is 0x2000 0001, the low two bits are 0b01, the effective length of the buffer is 2047 bytes whose address range is from 0x20000001 (for the first received frame byte) to 0x2000 07FF.

When a start of frame (SOF) is received, the FSG bit is set by DMA controller and when the end of the frame (EOF) is received, the LSG bit is set. If the receive buffer size is programmed to be large enough to store the whole frame, the FSG and the LSG bit are set in the same descriptor. The actual frame length FRML can be read from Receive Descriptor0. So application can calculate the left unused buffer space. The RxDMA always uses a new

descriptor to receive the start of next frame.

### Arbitration for TxDMA and RxDMA controller

There are two types of arbitration method designed for improving the efficiency of DMA controller between transmission and reception: fixed-priority and round-robin. When DAB bit in ENET_DMA_BCTL register is reset, arbiter selects round-robin method. The arbiter allocates the data bus in the ratio set by the RTPR bits in ENET_DMA_BCTL, when both of TxDMA and RxDMA controller request access simultaneously. When DAB bit in ENET_DMA_BCTL register is set, arbiter selects fixed-priority, and the RxDMA controller always has higher priority over the TxDMA.

### DMA error status

During the operation of the DMA controller, when a response error presents on the bus, the DMA controller considers a fatal error occurs and stops operating at once with error flags written to the DMA status register (ENET_DMA_STAT). After such fatal error (response error) occurs, application must reset the Ethernet module and reinitialize the DMA controller.

### DMA controller initialization for transmission and reception

Before using the DMA controller, the initialization must be done as follow steps:

1. Set the bus access parameters by writing the ENET_DMA_BCTL register;
2. Mask unnecessary interrupt source by configuring the ENET_DMA_INTEN register;
3. Program the Tx and Rx descriptor table start address by writing the ENET_DMA_TDTADDR register and the ENET_DMA_RDTADDR register;
4. Configure filter option by writing related registers;
5. According to the auto-negotiation result with external PHY, set the SPD bit and DPM bit for selecting the communication mode (Half-duplex / Full-duplex) and the communication speed (10Mbit / s or 100Mbit / s). Set the TEN and REN bit in ENET_MAC_CFG register to enable MAC transmit and receive operations;
6. Set STE bit and SRE bit in ENET_DMA_CTL register to enable TxDMA controller and RxDMA controller.

**Note:** If the HCLK frequency is too much low, application can enable RxDMA before set REN bit in ENET_MAC_CFG register to avoid RxFIFO overflow at start time.

### Transmit process of DMA

As mentioned before, a frame can span over several buffers which means several descriptors. When the FSG bit is set, the descriptor indicates the start of the frame and when the LSG bit is set, the descriptor indicates the end of the frame. All the buffers among these descriptors store the whole frame data. When the last descriptor is fetched and buffer finished reading, the transmitting status will write back to it. The other descriptors (here means the descriptor whose LSG bit is reset) of the current frame will not be changed by TxDMA controller except the DAV bit will be reset to 0. After starting transfer frame data from memory to FIFO, the

transmitting has not actually start. The real start time for sending frame on interface is depended on TxDMA mode: Cut-Through mode or Store-and-Forward mode. The former mode starts sending when the byte number of FIFO is greater than configured threshold and the latter mode starts sending when the whole frame data are transferred into FIFO or when the FIFO is almost full.

### Transmission management of DMA

#### Operate on second frame in buffer

When OSF bit in ENET_DMA_CTL is reset, the order of the transmitting is follows: the first is reading transmit descriptor, followed by reading data from memory and writing to FIFO, then sending frame data on interface through MAC and last wait frame data transmitting complete and writing back transmitting status.

Above procedure is TxDMA's standard transmitting procedure but when HCLK is much faster than TX_CLK, the efficiency of transmitting two frames will be greatly reduced.

To avoid the case mentioned above, application can set OSF to 1. If so, the second frame data can be read from the memory and push into FIFO without waiting the first frame's status writing back. OSF function is only performed between two neighboring frames.

#### TxDMA operation mode (A) (default mode): Non-OSF

The TxDMA controller in Non-OSF mode proceeds as follows:

1. Initialize the frame data into the buffer space and configure the descriptor (Transmit Descriptor0-3) with DAV bit of Transmit Descriptor0 sets to 1;
2. Enable TxDMA controller by setting STE bit in ENET_DMA_CTL register;
3. The TxDMA controller starts continue polling and performing transmit descriptor. When the DAV bit in Transmit Descriptor0 that TxDMA controller read is cleared, or any error condition occurs, the controller will enter suspend state and at the same time both the transmit buffer unavailable bit in ENET_DMA_STAT and normal interrupt summary bit in ENET_DMA_STAT register are set. If entered into suspend state, operation proceeds to Step 8;
4. When the DAV bit in Transmit Descriptor0 of the acquired descriptor is set, the DMA decodes the transmit frame configured and the data buffer address from the acquired descriptor;
5. DMA retrieve data from the memory and push it into the TxFIFO of MAC;
6. The TxDMA controller continues polling the descriptor table until the EOF data (LSG bit is set) is transferred. If the LSG bit of current descriptor is reset, it will be closed by resetting the DAV bit after all buffer data pushed into TxFIFO. Then the TxDMA controller waits to write back descriptor status and IEEE 1588 timestamp value if enabled;
7. After the whole frame is transferred, the transmit status bit (TS bit in ENET_DMA_STAT register) is set only when INTC bit in Transmit Descriptor0 is set. Also an interrupt generates if the corresponding interrupt enable flag is set. The TxDMA controller returns to Step 3 for the next frame;

8. In the suspend state, application can make TxDMA returns to running state by writing any data to ENET_DMA_TPEN register and clearing the transmit underflow flag. Then the TxDMA controller process turns to Step 3.

**TxDMA operation mode (B): OSF**

The TxDMA controller supports transmitting two frames without waiting status write back of the first frame, this mode is called operation on second frame (OSF). When the frequency of system is much faster than the frequency of the MAC interface (10Mbit / s or 100Mbit / s), the OSF mode can improve the sending efficiency. Setting OSF bit in ENET_DMA_CTL register can enable this mode. When the TxDMA controller received EOF of the first frame, it will not enter the state of waiting status write back but to fetch the next descriptor, if the DAV bit and FSG bit of the next descriptor is set, the TxDMA controller immediately read the second frame data a push them into the MAC FIFO.

The TxDMA controller in OSF mode proceeds as follows:

1. Follow steps 1-6 operation in TxDMA default mode;
2. The TxDMA controller retrieves the next descriptor without closing the previous frame's last descriptor in which the LSG bit is set;
3. If the DAV bit of the next descriptor is set, the TxDMA controller starts reading the next frame's data from the buffer address. If the DAV bit of the next descriptor is reset, TxDMA controller enters suspend state and the next operation goes to Step 7;
4. TxDMA controller continues polling descriptor and frame data until the EOF is transferred. If a frame is described with more than one descriptor, the intermediate descriptors are all closed by TxDMA controller after fetched;
5. The TxDMA controller enters the state of waiting for the transmission status and time stamp of the previous frame (if timestamp enabled). With writing back status to descriptor, the DAV bit is also cleared by TxDMA controller;
6. After the whole frame is transferred, the transmit status bit (TS bit in ENET_DMA_STAT register) is set only when INTC bit in Transmit Descriptor0 is set. Also an interrupt generates if the corresponding interrupt enable flag is set. The TxDMA controller returns to Step 3 for the next frame if no underflow error occurred in previous frame. If underflow error of the previous frame is occurred, the TxDMA controller enters in suspend state and the next operation goes to Step 7;
7. In suspend state, when the status information and timestamp value (if the function is enable) of the transmitting frame is available, the TxDMA controller writes them back to descriptor and then close it by setting DAV=0 of descriptor;
8. In suspend state, application can make TxDMA returns to running state by writing any data to ENET_DMA_TPEN register and clearing the transmit underflow flag. Then the TxDMA controller process goes to Step 1 or Step 2.

### Transmit frame format in buffer

According to IEEE 802.3 specification described before, a frame structure is made up of such fields: Preamble, SFD, DA, SA, QTAG (option), LT, DATA, PAD (option), and FCS.

The Preamble and SFD are automatically generated by the MAC, so the application only need store the DA, SA, QTAG (if needed), LT, DATA, DATA, PAD (if needed), FCS (if needed) parts. If the frame needs padding which means PAD and FCS parts are not stored in buffer, then application can configure the MAC to generate the PAD and FCS. If the frame only need FCS which means only FCS part is not stored in buffer, the application can configure the MAC to generate FCS. The DPAD bit and DCRC bit are designed to achieve the generate function of the PAD and FCS field.

### Suspend during transmit polling

The DMA controller keeps querying the transmit descriptor after the transmission is started. If either of the following conditions happens, the DMA controller will enter suspend state and the transmit polling will stop. Though the DMA entered suspend state, the descriptor pointer is maintained to the descriptor following of the last closed descriptor.

- The DMA controller fetches a descriptor with DAV=0, then it enters suspend state and stops polling. In this case, the NI bit and TBU bit in ENET_DMA_STAT register are set.
- The MAC FIFO is empty during sending a frame on interface which means an error of underflow occurs. In this case, the AI bit and TU bit in ENET_DMA_STAT register are set. Also the transmit error status will write back to transmit descriptor.

### Transmit DMA descriptor with IEEE 1588 timestamp format

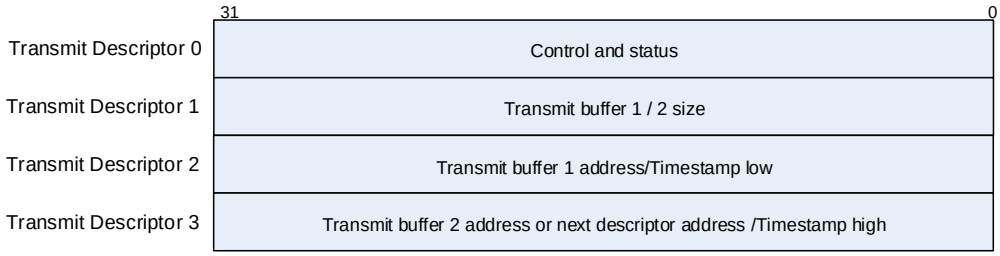
When TTSEN bit is set, the timestamp function is enabled. The TxDMA controller writes transmit timestamp status TTMSS and timestamp back to descriptor after the frame transmission complete. The word address in descriptor for writing timestamp is depends on DFM bit in ENET_DMA_BCTL register. If the descriptor format is normal mode (DFM=0), Transmit Descriptor2 and Transmit Descriptor3 are used for timestamp recording and the old values in Transmit Descriptor2 and Transmit Descriptor3 are overwritten. If the descriptor format is enhanced mode (DFM=1), Transmit Descriptor6 and Transmit Descriptor7 are used for timestamp recording and the value in Transmit Descriptor2 and Transmit Descriptor3 are kept.

### TxDMA descriptors in normal mode

The normal mode descriptor structure consists of four 32-bit words: Transmit Descriptor0 ~ Transmit Descriptor3**.** The descriptions of Transmit Descriptor0 ~ Transmit Descriptor3 are given below:

**Note:** When a frame is described by more than one descriptor, only the control bits of the first descriptor are accept by TxDMA controller (except INTC). But the status and timestamp (if enabled) are written back to the last descriptor.

**Figure 43-7. Transmit descriptor in normal mode**




■ Transmit Descriptor0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAV | INTC | LSG | FSG | DCRC | DPAD | TTSEN | Reserved | CM[1:0] | | TERM | TCHM | Reserved | | TTMSS | IPHE |
| rw | rw | rw | rw | rw | rw | rw | | rw | | rw | rw | | | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ES | JT | FRMF | IPPE | LCA | NCA | LCO | ECO | VFRM | COCNT[3:0] | | | | EXD | UFE | DB |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DAV | DAV bit<br>The DMA clears this bit either when it completes the frame transmission or the buffer allocated in the descriptor is read completely. This bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set.<br>0: The descriptor is available for CPU not for DMA<br>1: The descriptor is available for DMA not for CPU |
| 30 | INTC | Interrupt on completion bit<br>Only when the LSG bit is set, this bit is valid.<br>0: TS bit in ENET_DMA_STAT is not set when frame transmission complete.<br>1: TS bit in ENET_DMA_STAT is set when frame transmission complete. |
| 29 | LSG | Last segment bit<br>This bit shows whether the transmit buffer contains the last segment of the frame.<br>0: The buffer of descriptor is not stored the last part of frame<br>1: The buffer of descriptor is stored the last part of frame |
| 28 | FSG | First segment bit<br>This bit shows whether the buffer contains the first segment of a frame.<br>0: The buffer of descriptor is not stored the first block of frame<br>1: The buffer of descriptor is stored the first block of frame |
| 27 | DCRC | Disable CRC bit<br>Only when the FSG bit is set, this bit is valid.<br>0: Allow MAC to insert CRC at the end of transmitted frame automatically<br>1: Not Allow MAC to insert CRC at the end of transmitted frame |
| 26 | DPAD | Disable adding pad bit<br>Only when the FSG bit is set, this bit is valid. |

| | | |
|---|---|---|
| | | 0: The DMA adds padding byte and CRC to transmitted frame automatically. Only the padding actually acts, the CRC is also appended. And ignore the value of DCRC bit.<br>1: The MAC does not add padding to a frame automatically |
| 25 | TTSEN | Transmit timestamp function enable bit.<br>Only when the FSG bit is set, this bit is valid.<br>0: Disable transmit timestamp function<br>1: Enable IEEE 1588 hardware time stamping for the transmit frame, when TMSEN bit in the ENET_PTP_TSCTL register is set. |
| 24 | Reserved | Must be kept at reset value. |
| 23:22 | CM[1:0] | Checksum mode bits<br>0x0: Disable checksum insertion function<br>0x1: Only enable function for IP header checksum calculation and insertion<br>0x2: Enable IP header checksum and payload checksum calculation and insertion, hardware does not calculate checksum of pseudo-header.<br>0x3: Enable IP Header checksum and payload checksum calculation and insertion, hardware calculates checksum of pseudo-header. |
| 21 | TERM | Transmit end for ring mode bit<br>This bit is used only in ring mode and has higher priority than TCHM.<br>0: The current descriptor is not the last descriptor in the table<br>1: The descriptor table reached its final descriptor. The DMA descriptor pointer returns to the start address of the table. |
| 20 | TCHM | The second address chained mode bit<br>This bit is used only in chain mode. When TCHM bit is set, TB2S[12:0] is ignored.<br>0: The second address in the descriptor is the second buffer address<br>1:The second address in the descriptor is the next descriptor address |
| 19:18 | Reserved | Must be kept at reset value. |
| 17 | TTMSS | Transmit timestamp status bit<br>Only when the LSG bit is set, this bit is valid.<br>0: Timestamp was not captured<br>1:A timestamp was captured for the described transmit frame and push into Transmit Descriptor2 (or Transmit Descriptor6 if DFM=1) and Transmit Descriptor 3 (or Transmit Descriptor7if DFM=1) |
| 16 | IPHE | IP header error bit<br>IP header error occurs when any case of below happen:<br>IPv4 frames:<br>1) The header length field has a value less than 0x5.<br>2) The header length field value in transmitting IPv4 frame is mismatch with the number of header bytes.<br>3) The version field value does not match the length / type field value. |

| | | |
|---|---|---|
| | | IPv6 frames:<br>1) The main header length is not 40 bytes.<br>2) The version field value does not match the length / type field value.<br>0: The MAC transmitter did not detect error in the IP datagram header<br>1: The MAC transmitter detected an error in the IP datagram header |
| 15 | ES | Error summary bit<br>Following bits are logical ORed to generate this bit:<br>IPHE: IP header error<br>JT: Jabber timeout<br>FRMF: Frame flush<br>IPPE: IP payload error<br>LCA: Loss of carrier<br>NCA: No carrier<br>LCO: Late collision<br>ECO: Excessive collision<br>EXD: Excessive deferral<br>UFE: Underflow error |
| 14 | JT | Jabber timeout bit<br>Only set when the JBD bit is reset.<br>0: No jabber timeout occurred<br>1: Jabber timeout of MAC transmitter has occurred |
| 13 | FRMF | Frame flushed bit<br>This bit is set to flush the Tx frame by software. |
| 12 | IPPE | IP payload error bit<br>The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP or ICMP packet bytes received from the application and issues an error status in case of a mismatch.<br>0: No IP payload error occurred<br>1: MAC transmitter detected an error in the TCP, UDP, or ICMPIP datagram payload. |
| 11 | LCA | Loss of carrier bit<br>When the interface signal ‘CRS’ lost one or more cycles and no collision happened during transmitting, the loss of carrier condition occurs.<br>Only in Half-duplex mode this bit is valid.<br>0: No loss of carrier occurred<br>1: When the frame is transmitting, loss of carrier occurred |
| 10 | NCA | No carrier bit<br>0: PHY carrier sense signal is active<br>1: When the frame is transmitting, the carrier sense signal from the PHY was not active |
| 9 | LCO | Late collision bit |

| | | |
|---|---|---|
| | | If a collision occurs when 64 bytes (including preamble and SFD) has already transferred, this situation called late collision.<br>0: No late collision occurred<br>1: Late collision situation occurred<br>**Note:** This bit is not valid if the UFE bit is set. |
| 8 | ECO | Excessive collision bit<br>If the RTD=1 (retry function disable), this bit is set after the first collision.<br>If the RTD=0 (retry function enable), this bit is set when failed 16 successive retry transmitting.<br>When this bit is set, the transmission of current frame is aborted.<br>0: No excessive collision occurred<br>1: Excessive collision occurred |
| 7 | VFRM | VLAN frame bit<br>0: The transmitted frame was a normal frame<br>1: The transmitted frame was a VLAN-type frame |
| 6:3 | COCNT[3:0] | Collision count bits<br>Only when ECO bit is cleared, this bit is valid.<br>Before the frame was transmitted, this 4-bit counter counts the number of collisions that has occurred. |
| 2 | EXD | Excessive deferral bit<br>Only when the DFC bit in the ENET_MAC_CFG register is set, this bit is valid.<br>0: No excessive deferral occurred<br>1: The transmission has ended because of excessive deferral time is over 3036 bytes |
| 1 | UFE | Underflow error bit<br>This bit shows that the TxDMA comes across an empty TxFIFO while transmitting the frame before EOF which is caused by pushing data to TxFIFO late from memory. The transmission process enters the suspend state and sets both the TU (bit 5) and the TS (bit 0) in ENET_DMA_STAT.<br>0: No underflow error occurred<br>1: Underflow error occurred and the MAC aborted the frame transmitting |
| 0 | DB | Deferred bit<br>This bit shows whether the transmitting frame is deferred because of interface signal CRS is active before MAC transmit frame.<br>Only in Half-duplex mode this bit is valid.<br>0: No transmission deferred<br>1:The MAC is deferred before transmission |

- Transmit descriptor1

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | TB2S[12:0] | | | | | | | | | | | | |

rw

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | TB1S[12:0] | | | | | | | | | | | | |

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:16 | TB2S[12:0] | Transmit buffer 2 size bits<br>The second data buffer byte size. If the THCM bit in the Transmit Descriptor0 is set, this bit is ignored. |
| 15:13 | Reserved | Must be kept at reset value. |
| 12:0 | TB1S[12:0] | Transmit buffer 1 size bits<br>The first data buffer byte size. The TxDMA ignores this buffer and uses buffer 2 (for TCHM=0) or the next descriptor (for TCHM=1) when this field is 0. |

- Transmit descriptor2

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TB1AP / TTSL[31:16] | | | | | | | | | | | | | | | |

rw

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TB1AP / TTSL[15:0] | | | | | | | | | | | | | | | |

rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TB1AP / TTSL[31:0] | Transmit buffer 1 address pointer / Transmit frame timestamp low 32-bit value bits<br>Before transmitting frame, application must configure these bits for transmit buffer 1 address (TB1AP). When the transmitting frame is complete, these bits can be changed to the timestamp low 32-bit value (TTSL) for transmitting frame if DFM=0. But if DFM=1, these bits will not change and keep the value of buffer address. When these bits stand for buffer 1 address (TB1AP), the alignment is no limitation. When these bits stand for timestamp low 32-bit value, the TTSEN and LSG bit of current descriptor must be set. |

- Transmit descriptor word 3

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TB2AP / TTSH[31:16] | | | | | | | | | | | | | | | |

rw

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TB2AP / TTSH[15:0] | | | | | | | | | | | | | | | |

rw

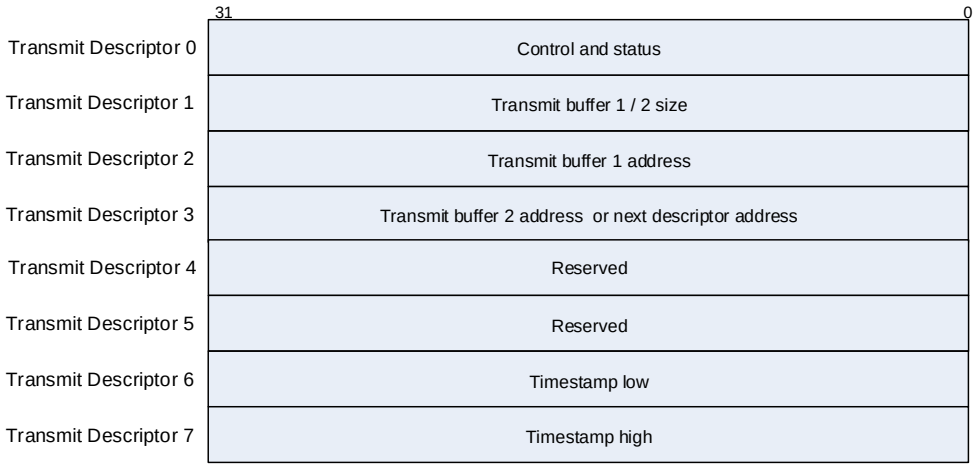
| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TB2AP / TTSH[31:0] | Transmit buffer 2 address pointer (or next descriptor address) / Transmit frame timestamp high 32-bit value bits.<br>Before transmitting frame, application must configure these bits for transmit buffer 2 address (TB2AP) or the next descriptor address which is decided by descriptor type is ring or chain. When the transmitting frame is complete, these bits can be changed to the timestamp high 32-bit value (TTSH) for transmitting frame if DFM=0 and TTSEN =1. But if DFM=1 or TTSEN =0, these bits will not change and keep the old value. When these bits stand for buffer 2 address (TCHM=0), the alignment is no limitation. When these bits stand for the next descriptor address (TCHM=1), these bits must be word-alignment. When these bits stand for timestamp high 32-bit value, the TTSEN and LSG bit of current descriptor must be set. |

### TxDMA descriptors in enhanced mode

The enhanced mode descriptor structure consists of eight 32-bit words: Transmit Descriptor0 ~ Transmit Descriptor7. The descriptions of Transmit Descriptor0 ~ Transmit Descriptor3. are the same with normal mode descriptor; Transmit Descriptor4 ~ Transmit Descriptor7. are given below:

**Note:** When a frame is described by more than one descriptor, only the control bits of the first descriptor are accept by DMA controller (except INTC). But the status and timestamp (if enabled) are written back to the last descriptor.

**Figure 43-8. Transmit descriptor in enhanced mode**




- Transmit descriptor4
  All bits reserved.
- Transmit descriptor5
  All bits reserved.
- Transmit descriptor6

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TTSL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TTSL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TTSL[31:0] | Transmit frame timestamp low 32-bit value bits<br>When TTSEN =1 and LSG=1, there bits are updated by TxDMA for recording timestamp low 32-bit value of the current transmitting frame. |

- Transmit descriptor 7

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TTSH[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TTSH[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TTSH[31:0] | Transmit frame timestamp high 32-bit value bits<br>When TTSEN =1 and LSG=1, these bits are updated by TxDMA for recording timestamp high 32-bit value of the current transmitting frame. |

### Reception process of DMA

When a frame is presented on the interface, the MAC starts to receive it. At the same time, the address filter block is running for this received frame. If the received frame fails the address filtering it will be discarded from RxFIFO in MAC and not be forwarded to buffer by RxDMA controller. If the received frame passes the address filtering, it will be forwarded to buffer when the available time comes. If the RxDMA controller is configured in Cut-Through mode, the available time means the byte number of the received frame is equal or greater than the configured threshold. If the RxDMA controller is configured in Store-and-Forward mode, the available time means the complete frame is stored in RxFIFO. During receiving frame, if any one of the below cases occurs the MAC can discard the received frame data in RxFIFO and the RxDMA controller will not forward these data:

- The received frame bytes is less than 64.
- Collision occurred during frame receiving.
- The premature termination for the receiving frame.

When the available time comes, the RxDMA controller starts transfer frame data from RxFIFO to the receive buffer. If the SOF is included in current receive buffer, the FDES bit in Receive

Descriptor0 is set when the RxDMA controller writing receive frame status to indicate this descriptor is used for storing the first part of the frame. If the EOF is included in current receive buffer, the LDES bit in Receive Descriptor0 is set when RxDMA controller writing receive frame status to indicate this descriptor is used for storing the last part of the frame. Often when the buffer size is larger than received frame, the FDES and LDES bit are set in the same descriptor. When the EOF is transferred to buffer or the receive buffer space is exhausted, the RxDMA controller fetches the next receive descriptor and closes previous descriptor by writing Receive Descriptor0 with DAV=0. If the LDES bit is set, the other status are also be updated and the RS bit in ENET_DMA_STAT register will be set (immediately when DINTC=0 or delayed when DINTC=1). If the DAV bit of the next descriptor is set, the RxDMA controller repeats above operation when received a new frame. If the DAV bit of the next descriptor is reset, the RxDMA controller enters suspend state and sets RBU bit in ENET_DMA_STAT register. The pointer value of descriptor address table is retained and be used for the starting descriptor address after exiting suspend state.

### Reception management of DMA

The receiving process of the RxDMA controller is described detailed as below:

1. Applications initialize the receive descriptors with the DAV bit in the Receive Descriptor0 is set;
2. Setting the SRE bit in ENET_DMA_CTL register to make RxDMA controller entering running state. In running state, the RxDMA controller continually fetching the receive descriptors from descriptor table whose starting address is configured in ENET_DMA_RDTADDR register by application. If the DAV bit of the fetched receive descriptor is set, then this descriptor is used for receiving frame. But if the DAV bit is reset which means this receive descriptor cannot be used by RxDMA, the RxDMA controller will enter suspend state and operation goes to Step 9;
3. From the valid receive descriptor (DAV=1), the RxDMA controller marks the receiving control bit and data buffer address;
4. Processing the received frames and transfer data to the receive buffer from the RxFIFO;
5. If all frame data has completely transferred or the buffer is full, the RxDMA controller fetches the next descriptor from receive descriptor table;
6. If the current receiving frame transfer is complete, the operation of RxDMA goes to Step7. But if not complete, two conditions may occur:
   - The next descriptor's DAV bit is reset. The RxDMA controller sets descriptor error bit DERR in Receive Descriptor0 if flushing function is enabled. The RxDMA controller closes current descriptor by resetting DAV bit and sets the LSG bit (if flushing is enabled) or resets the LSG bit (if flushing is disabled). Then the operation goes to Step 8.
   - The next descriptor's DAV bit is set. The RxDMA controller closes current descriptor by resetting DAV bit and operation goes to Step 4.
7. If IEEE 1588 time stamping function is enabled, the RxDMA controller writes the time stamp value (if receiving frame meets the configured time stamping condition) to the

current descriptor's Receive Descriptor2 and Receive Descriptor3 if DFM=0 or Receive Descriptor6 and Receive Descriptor7 if DFM=1. At the same time (writing timestamp value) the RxDMA controller also writes the received frame's status word to the Receive Descriptor0 with the DAV bit cleared and the LSG bit set;

8. The latest descriptor is fetched by RxDMA controller. If the fetched descriptor bit 31 (DAV) is set, the RxDMA controller operation goes to Step 4. If the fetched descriptor bit 31 is reset, the RxDMA controller enters the suspend state and sets the RBU bit in register ENET_DMA_STAT. If flushing function is enabled, the RxDMA controller will flush the received frame data in the RxFIFO before entering suspend state;
9. In suspended state, there are two conditions to exit. The first is writing data in the ENET_DMA_RPEN register by application. The second is when a new received frame is available which means the byte number of receiving frame is greater than threshold in Cut-Through mode or when the whole frame is received in Store-and-Forward mode. Once exiting suspend mode, the RxDMA controller fetches the next descriptor and the following operation goes to Step 2.

### Receive descriptor fetching regulation

Descriptor fetching occurs if any one or more of the following conditions are met:

- The time SRE bit is configured from 0 to 1 which makes the RxDMA controller entering running state.
- The total buffer size (buffer 1 for chain mode or buffer 1 plus buffer 2 for ring mode) of the current descriptor cannot hold the current receiving frame. In other word, the last byte stored in buffer space is not the EOF byte.
- After a complete frame is transferred to buffer and before current descriptor is closed.
- In suspend state, the MAC received a new frame.
- Writing any value to receive poll enable register ENET_DMA_RPEN.

### Processing after a new frame received in suspend state

When a new frame is available (see available definition in the previous paragraph), the RxDMA controller fetches the descriptor. If the DAV bit in Receive Descriptor0 is set, the RxDMA controller exits suspend state and returns to running state for frame reception. But if the DAV bit in Receive Descriptor0 is reset, application can choose whether these received frame data in RxFIFO are flushed or not by configuring DAFRF bit in ENET_DMA_CTL register. If DAFRF=0, the RxDMA controller discards these received frame data and makes the missed frame counter (MSFC) increase one. If DAFRF=1, these frame data are will not be flushed and MSFC counter will not increase until the RxFIFO is full. If the DAV bit is reset in fetched descriptor, the RBU bit in ENET_DMA_STAT register will be set and the RxDMA controller will be still in suspend state.

### Receive DMA descriptor with IEEE 1588 timestamp format

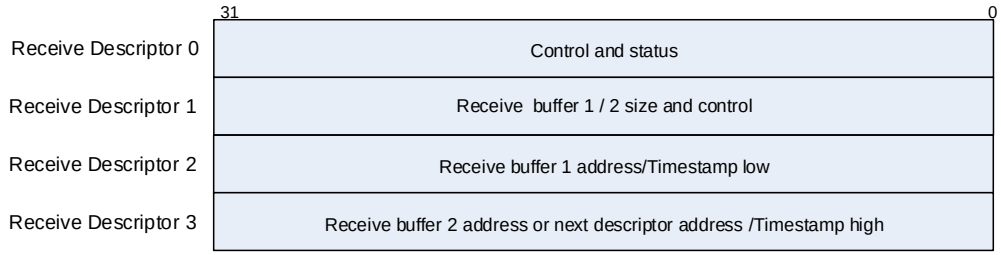
If the IEEE 1588 function enabled, the MAC writes the timestamp value to Receive Descriptor2 and Receive Descriptor3 (DFM=0) or Receive Descriptor6 and Receive

Descriptor7 (DFM=1) after a frame with timestamp reception complete and before the RxDMA controller clears the DAV bit.

### RxDMA descriptors in normal mode

In normal descriptor mode, the descriptor structure consists of four 32-bit words: Receive Descriptor0 ~ Receive Descriptor3 The detailed description of Receive Descriptor0 ~ Receive Descriptor3 are given below.

**Figure 43-9. Receive descriptor in normal mode**




■ Receive descriptor 0

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAV | DAFF | FRML[13:0] | | | | | | | | | | | | | |
| rw | rw | rw | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERRS | DERR | SAFF | LERR | OERR | VTAG | FDES | LDES | IPHERR / TSV | LCO | FRMT | RWDT | RERR | DBERR | CERR | PCERR / EXSV |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DAV | Descriptor available bit<br>This bit shows the DMA controller can use this descriptor. The DMA clears this bit either when it completes the frame reception or when the buffers in this descriptor are full.<br>0: The descriptor is owned by the CPU<br>1: The descriptor is owned by the DMA |
| 30 | DAFF | Destination address filter fail bit<br>0: A frame passed the destination address filter<br>1: A frame failed the destination address filter |
| 29:16 | FRML[13:0] | Frame length bits<br>These bits show the byte length of the received frame that was transferred to the buffer (including CRC when received frame is not a type frame. If received frame is a type frame, including CRC or not is controlled by TFCD bit in ENET_MAC_CFG). Only when the bit LDES=1 and DERR=0, these bits are valid. If LDES=0 and ERRS=0, these bits indicate the accumulated number of bytes that have been transferred for the current frame. |

**Note:** The value of frame length is 0 means that for some reason (such as FIFO overflow or dynamically modify the filter value in the receiving process, resulting did not pass the filter, etc), frame data is not written to FIFO completely.

| | | |
|---|---|---|
| 15 | ERRS | Error summary bit<br>Only when the LDES bit is set, this bit is valid.<br>This bit is logical ORed by the following bits when DFM is equal to 0:<br>DERR: Descriptor error<br>OERR: Overflow error<br>LCO: Late collision<br>RWDT: Watchdog timeout<br>RERR: Receive error<br>CERR: CRC error<br>IPHERR = 0, FRMT = 1 and PCERR = 1: payload checksum error<br>IPHERR = 1, FRMT = 1 and PCERR = 0: header checksum error<br>IPHERR = 1, FRMT = 1 and PCERR = 1: both header and payload checksum errorsThis bit is logical ORed by the following bits when DFM is equal to 1:<br>IPPLDERR: IP frame payload error<br>IPHERR: IP frame header error<br>DERR: Descriptor error<br>OERR: Overflow error<br>LCO: Late collision<br>RWDT: Watchdog timeout<br>RERR: Receive error<br>CERR: CRC error |
| 14 | DERR | Descriptor error bit<br>Only when the LDES bit is set, this bit is valid.<br>When the current buffer cannot hold current received frame and the next descriptor's DAV bit is reset, the descriptor error occurs.<br>0: No descriptor error occurred<br>1: Descriptor error occurred |
| 13 | SAFF | SA filtering fail bit<br>0: No source address filter fail occurred<br>1: A received frame failed the SA filter |
| 12 | LERR | Length error bit<br>Only when the FRMT bit is reset, this bit is valid.<br>This bit shows whether the length field in received is mismatch the actual frame length.<br>0: No length error occurred<br>1: Length error occurred |
| 11 | OERR | Overflow error bit |

| | | |
|---|---|---|
| | | When RxFIFO is overflow and the frame data has been partly forwarded to descriptor buffer, the overflow error bit sets.<br>0: No overflow error occurred<br>1: RxFIFO overflowed and frame data is not valid |
| 10 | VTAG | VLAN tag bit<br>0: Received frame is not a tag frame<br>1: Received frame is a tag frame |
| 9 | FDES | First descriptor bit<br>This bit shows whether current descriptor contains the SOF of the received frame.<br>0: The current descriptor does not store the SOF of the received frame<br>1: The current descriptor buffer saves the SOF of the received frame |
| 8 | LDES | Last descriptor bit<br>This bit shows whether current descriptor contains the EOF of the received frame.<br>0: The current descriptor buffer does not store EOF of the received frame<br>1: The current descriptor buffer saves the EOF of the received frame |
| 7 | IPHERR / TSV | IP frame header error bit / Timestamp valid bit<br>When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table.<br>When DFM=1, this bit indicates the timestamp value is taken and write to the Receive Descriptor6 and Receive Descriptor7. This bit is valid only when LDES is set. |
| 6 | LCO | Late collision bit<br>This bit shows whether a collision occurs after 64 bytes have been received.<br>This bit only valid in Half-duplex mode.<br>0: No late collision occurred<br>1: Late collision has occurred |
| 5 | FRMT | Frame type bit<br>When DFM=0, bit 7, 5 and 0 shows some special cases refer to the error status table.<br>When DFM=1, this bit shows the received frame is an Ethernet type frame or a tagged frame.<br>If the received frame is runt frame, this bit is not valid for application.<br>0: The received frame is an IEEE802.3 frame without tagged.<br>1: The received frame is an Ethernet-type frame (the length / type field is greater than or equal to 0x0600, or this is a tagged frame) |
| 4 | RWDT | Receive watchdog timeout bit<br>When WDD=0, this bit shows a frame with more than 2048 bytes was detected.<br>When WDD=1, this bit shows a frame with more than 16384 bytes was detected.<br>0: No receive watchdog timeout occurred |

1: Watchdog timer overflowed during receiving and current frame is only a part of frame.

3 RERR
Receive error bit
This bit shows whether the interface signal RX_ER asserted when RX_DV signal is active during frame receiving process.
0: No receive error occurred
1:Receive error occurred

2 DBERR
Dribble bit error bit
This bit shows whether there is an incomplete byte (odd cycles during reception) received. Only when in MII interface mode, this bit is valid.
0: No dribble bit error occurred
1: Dribble bit error occurred

1 CERR
CRC error bit
This bit shows whether FCS field in received frame is mismatch with the calculation result of the hardware. Only when LDES bit is set, this bit is valid.
0: No CRC error occurred
1:A CRC error occurred

0 PCERR / EXSV
Payload checksum error bit / Extended status valid bit
When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table.
When DFM=1, this bit indicates the descriptor Receive Descriptor4is valid for application.
This bit only valid when LDES is set.
0: Receive Descriptor4is not valid for application
1: Receive Descriptor4is valid for application

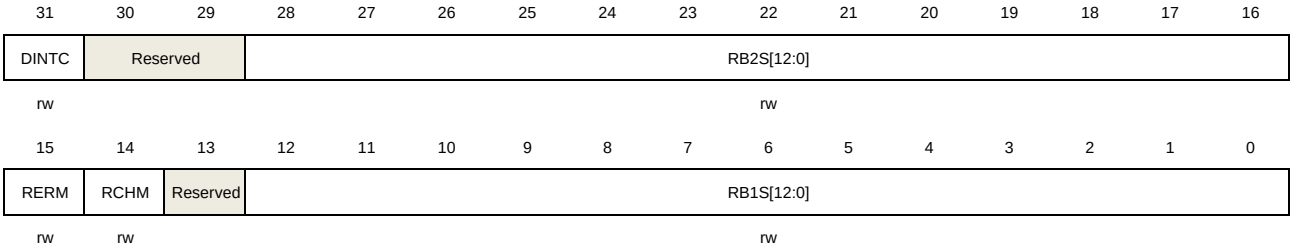
*Table 43-7. Error status decoding in Receive Descriptor0, only used for normal descriptor (DFM=0)* shows the combination meaning for bit 7, 5, and 0 in Receive Descriptor0:

**Table 43-7. Error status decoding in Receive Descriptor0, only used for normal descriptor (DFM=0)**

| Bit 7: IPHERR | Bit 5: FRMT | Bit 0: PCERR | Frame status |
|---|---|---|---|
| 0 | 0 | 0 | IEEE 802.3 normal frame (Length field value is less than 0x0600 and not tagged) |
| 0 | 0 | 1 | IPv4 or IPv6 frame, no header checksum error, payload checksum is bypassed because of unsupported payload type |
| 0 | 1 | 0 | IPv4 or IPv6 frame, checksum checking pass |
| 0 | 1 | 1 | IPv4 or IPv6 frame, payload checksum error.<br>This error may case by following condition:<br>1) Calculated checksum value mismatch the checksum field<br>2) byte number of received payload mismatch length field |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | Reserved |
| 1 | 0 | 1 | A type (length / type field equal or greater than 0x0600) or tagged frame but neither IPv4 nor IPv6.<br>Offload check engine is bypassed. |
| 1 | 1 | 0 | IPv4 or IPv6 frame, but a header checksum error detected<br>This error may case by following condition:<br>1) Type value inconsistent with version value<br>2) Calculated header checksum mismatch the header checksum field<br>3) Expected IP header bytes is not received enough |
| 1 | 1 | 1 | IPv4 or IPv6 frame, both header and payload checksum detected errors |

■ Receive descriptor 1




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DINTC | Disable interrupt on completion bit<br>0: RS bit in ENET_DMA_STAT register will immediately set after receiving the completed, then if enabled the corresponding interrupt, the interrupt will trigger.<br>1: RS bit in ENET_DMA_STAT register will is not immediately set after receiving the completed, but will set after a configurable delay time. |
| 30:29 | Reserved | Must be kept at reset value. |
| 28:16 | RB2S[12:0] | Receive buffer 2 size bits<br>The second buffer size in bytes. The buffer size must be a multiple of 4. This field is ignored if RCHM is set. |
| 15 | RERM | Receive end of ring mode bit<br>This bit indicates the final descriptor in table is arrived and the next descriptor address is automatically set to the configured start descriptor address.<br>0: Current descriptor is not the last descriptor in table<br>1: Current descriptor is the last descriptor in table |
| 14 | RCHM | Receive chained mode for second address bit<br>0: The second address points to the second buffer address.<br>1: The second address points to the next descriptor address. RB2S[12:0] is ignored. |

**Note:** If the RERM=1, the next descriptor returns to base address even this bit is set to 1.

| | | |
|---|---|---|
| 13 | Reserved | Must be kept at reset value. |
| 12:0 | RB1S[12:0] | Receive buffer 1 size bits<br>The first buffer size in bytes. The buffer size must be a multiple of 4. If this field is 0, the RxDMA controller ignores this buffer and uses buffer 2 (RCHM=0) or the next descriptor (RCHM=1). |

■ Receive descriptor 2

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RB1AP / RTSL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RB1AP / RTSL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RB1AP / RTSL[31:0] | Receive buffer 1 address pointer / Receive frame timestamp low 32-bit<br>These bits are designed for two different functions: buffer address pointer (RB1AP) or timestamp low 32-bit value (RTSL).<br>RB1AP: Before fetching this descriptor by RxDMA controller, these bits are configured to the buffer 1 address by application. This buffer 1 address pointer is used for RxDMA controller to store the received frame if RB1S is not 0. The buffer address alignment has no limitation.<br>RTSL: When timestamp function is enabled and LDES is set, these bits will be changed to timestamp low 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshot condition. If the received frame does not meet the snapshoot condition, these bits will keep RB1AP value. |

■ Receive descriptor 3

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RB2AP / RTSH[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RB2AP / RTSH[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RB2AP / RTSH[31:0] | Receive buffer 2 address pointer (next descriptor address) / Receive frame timestamp high 32-bit value bits<br>These bits are designed for two different functions: buffer address pointer or next descriptor address (RB1AP) or timestamp high 32-bit value (RTSH). |

RB2AP: Before fetching this descriptor by RxDMA controller, these bits are configured to the buffer 2 address (RCHM=0) or the next descriptor address (RCHM=1) by application. This buffer 2 address pointer is used for RxDMA controller to store the received frame if RB1S is not 0 when RCHM=0. If RCHM=1 and RERM=0, this address pointer is used for fetching the next descriptor. If RCHM=1 and RERM=1, these bits are ignored.

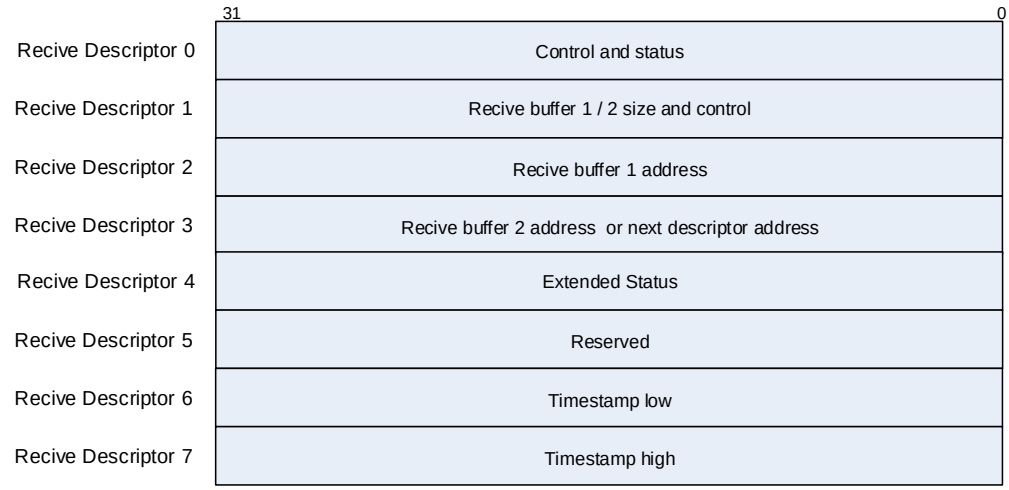
When this address is used for next descriptor address, the word alignment is needed. The other conditions have no limitation for these bits.

RTSH: When timestamp function is enabled and LDES is set, these bits will be changed to timestamp high 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition. If the received frame does not meet the snapshot condition, these bits will keep RB2AP value.

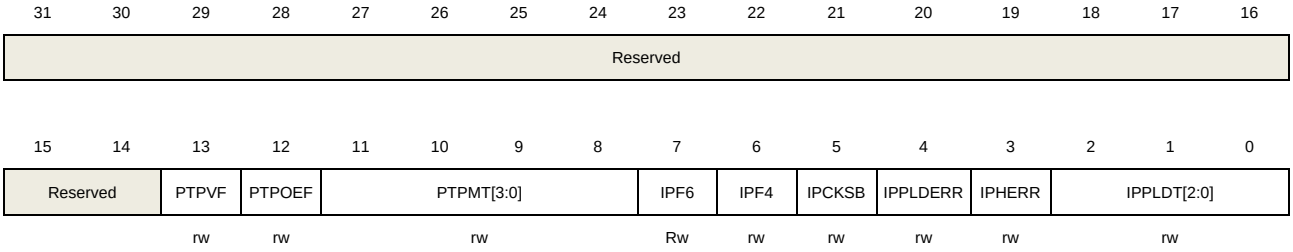
### RxDMA descriptors in enhanced mode

In enhanced descriptor mode, the descriptor structure consists of eight 32-bit words: Receive Descriptor0 ~ Receive Descriptor7. The description of Receive Descriptor0 ~ Receive Descriptor3 are the same with descriptors in normal mode. The description of Receive Descriptor4 ~ Receive Descriptor7 are given below.

**Figure 43-10. Receive descriptor in enhanced mode**




■ Receive descriptor 4




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 13 | PTPVF | PTP version format bit<br>0: Version 1 format<br>1: Version 2 format |
| 12 | PTPOEF | PTP on Ethernet frame bit<br>0: Received PTP frame is a IP-UDP frame if PTPMT is not zero<br>1: Received PTP frame is a IEEE802.3 Ethernet frame |
| 11:8 | PTPMT[3:0] | PTP message type bits<br>PTP message type is decoded to following number:<br>0x0: Not PTP frame received<br>0x1: SYNC<br>0x2: FOLLOW_UP<br>0x3: DELAY_REQ<br>0x4: DELAY_RESP<br>0x5: For peer-to-peer transparent clock: PDELAY_REQ<br>For ordinary or boundary clock: ANNOUNCE<br>0x6: For peer-to-peer transparent clock: PDELAY_RESP<br>For ordinary or boundary clock: MANAGEMENT<br>0x7: For peer-to-peer transparent clock: PDELAY_RESP_FOLLOW_UP<br>For ordinary or boundary clock: SIGNALING |
| 7 | IPF6 | IP frame in version 6 bit<br>0: Received frame is not a IPv6 frame<br>1: Received frame is a IPv6 frame |
| 6 | IPF4 | IP frame in version 4 bit<br>0: Received frame is not a IPv4 frame<br>1: Received frame is a IPv4 frame |
| 5 | IPCKSB | IP frame checksum bypassed bit<br>This bit is only valid when received frame is a IPv4 or IPv6 frame.<br>0: Received frame checksum checking function is not bypassed<br>1: Received frame checksum checking function is bypassed |
| 4 | IPPLDERR | IP frame payload error bit<br>This bit can be set by any of below cases: 1) the calculated checksum by hardware mismatch with the TCP, UDP or ICMP checksum field in frame. 2) payload length value in IP header mismatch the received payload length.<br>0: Payload error not occurred in received frame<br>1: Payload error occurred in received frame |
| 3 | IPHERR | IP frame header error bit<br>This bit can be set by any of below cases: 1) the calculated checksum by hardware mismatch with the IP header checksum field value. 2) Type field in IP frame is not consistent with version field (e.g. ‘type’ field value is 0x0800 but ‘version’ field value is not 0x4, ‘type’ field value is 0x86dd but ‘version’ field value is not 0x6).<br>0: IP header error not occurred |

1: IP header error occurred

| | | |
|---|---|---|
| 2:0 | IPPLDT[2:0] | IP frame payload type bits<br>These bits are valid only when IPFCO=1, IPHERR=0 and LDES=1.<br>0x0: Unsupported payload type or IP payload bypassed<br>0x1: payload type is UDP<br>0x2: payload type is TCP<br>0x3: payload type is ICMP<br>0x4~0x7: Reserved |

- Receive descriptor 5

  All bits reserved

- Receive descriptor 6

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTSL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RTSL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RTSL[31:0] | Receive frame timestamp low 32-bit value<br>When timestamp function is enabled and LDES is set, these bits will be written to timestamp low 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition. |

- Receive descriptor 7

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTSH[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RTSH[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RTSH[31:0] | Receive frame timestamp high 32-bit value<br>When timestamp function is enabled and LDES is set, these bits will be written to timestamp high 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshoot condition. |

### 43.3.4. MAC statistics counters: MSC

For knowing the statistics situation of transmitting and receiving frames, there is a group of

counters designed for gathering statistics data. These MAC counters are called statistics counters (MSC).In Section '***Register definition***', there is a detailed description of the function of these registers.

- When the transmit frame does not appear the situations, such as frame underflow, no carrier, carrier lost, excessive deferral, late collision， excessive collision and jabber timeout, it can be called "good frame". MSC transmit counters will automatically update.
- When the receiving frame does not appear the situations, such as alignment error, CRC mismatch , runt frame, length error, range error and error signal valid on pin MII_RX_ER, it can be called 'good frame' and MSC reception counters will automatically update. Among them, CRC mismatch indicates that calculated CRC value is different from FSC field value, runt frame indicates that the frame length is shorter than 64 bytes, length error indicates that the length field value is different from the actual received data bytes, range error indicates that the length field value is larger than maximum size of defined in IEEE802.3 (1518 for untagged frame and 1522 for VLAN tagged frame).

**Note:** Only when the discarded frame is a short frame whose length is less than 6 bytes (no complete receives the DA), MSC reception counter is updated.

### 43.3.5. Wake up management: WUM

Ethernet (ENET) module supports two wakeup methods from Deep-sleep mode. The one is remote wakeup frame and the other is Magic Packet wakeup frame. For reduce power consuming, the host system and Ethernet can be powered down and thus the circuit driven by HCLK or transmit clock is stop working. But the circuit driven by receive clock will continues working for listening wakeup frame. If application sets the PWD bit in ENET_MAC_WUM register, the Ethernet enters into power-down state. In power-down state, MAC ignores all the frame data on the interface until the power-down state is exited. For exiting power-down state, application can choose one of or both of the two methods mentioned above. Setting WFEN bit in ENET_MAC_WUM register can make Ethernet wakeup if a remote wakeup frame received and setting MPE bit in ENET_MAC_WUM register can make Ethernet wakeup if a Magic Packet frame is received. When any type of wakeup frame is present on interface and corresponding wakeup function is enabled, Ethernet will generate a wakeup interrupt and exit power-down state at once.

#### Remote wakeup frame detection

Setting WFEN bit in ENET_MAC_WUM register can enable remote wakeup detection. When the MAC is in power-down state and remote wakeup function enable bit is set, MAC wakeup frame filter is active. If the received frame passes the address filter and filter CRC-16 matches the incoming examined pattern, then MAC identified the received wakeup frame, and then MAC returns to normal working state. Even if the length of the wakeup frame exceeds 512 bytes, as long as the frame has a correct CRC value, it is still considered to be effective. After received the remote wakeup frame, the WUFR bit in ENET_MAC_WUM register will be set. If remote wakeup interrupt is not masked, then a WUM interrupt is generated.

### Magic packet detection

Another wakeup method is detecting Magic Packet frame (see 'Magic Packet Technology', Advanced Micro Devices). A Magic Packet frame is a special frame with formed packet solely intended for wakeup purposes. This packet can be received, analyzed and recognized by the Ethernet block and used to trigger a wakeup event. Setting MPE bit in ENET_MAC_WUM register can enable this function. This type of frame's format is as follows: starts by 6 continuous bytes of the value 0xFF (0xFFFF FFFF FFFF) in anywhere of the frame behind the destination and source address field, then there are 16 duplicate MAC addresses without any interruption and pause. If there is any discontinuity between repeating it 16 times, MAC needs to re-detect 0xFFFF FFFF FFFF in the receive frame. WUM module continuously monitors each frame received. When a Magic Packet frame passing the address filter, MAC will detect its format with Magic Packet format, once the format is matched the WUM will make MAC wakeup from power down state. Then the MAC wakes up from power-down state after receiving a Magic Packet frame. Module also accepts multicast frames as Magic Packet frame.

Example: An example of a Magic Packet with station address 0xAABB CCDD EEFF is the following (MISC indicates miscellaneous additional data bytes in the packet):

<DESTINATION><SOURCE><MISC>

…… FF FF FF FF FF FF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

<MISC><FCS>

Upon detecting a Magic Packet, the MPKR bit in ENET_MAC_WUM register will be set. If the Magic Packet interrupt is enabled, the corresponding interrupt will generate.

### Precautions during system power-down state

When the MCU is in Deep-sleep mode, if external interrupt line 22 (ENET1) / 23 (ENET0) is enabled, Ethernet WUM module can still detecting frames. Because the MAC in power-down state needs detecting Magic Packet or remote wakeup frame, the REN bit in ENET_MAC_CFG register must be maintained set. The transmit function should be turned disable during the power-down state by clearing the TEN bit in the ENET_MAC_CFG register. Moreover, the Ethernet DMA block should be disabled during the power-down state, because it is not necessary that the Magic Packet or remote wakeup frame is forwarded to the application. Application can disable the Ethernet DMA block by clearing the STE bit and the SRE bit (for the TxDMA and the RxDMA, respectively in the ENET_DMA_CTL register.

Follow steps are recommended for application to enter and exit power-down state:

1. Wait the current sending frame completes and then reset the TxDMA block by clearing STE bit in ENET_DMA_CTL register;
2. Clear the TEN and REN bit in ENET_MAC_CFG register to disable the MAC's transmit and receive function;
3. Check the RS bit in ENET_DMA_STAT register, waiting receive DMA read out all the frames in the receive FIFO and then close RxDMA;
4. Configure and enable the external interrupt line 19, so that it can generate an interrupt or event. If EXTI line 19 is configured to generate an interrupt, application still needs to modify ENET_WKUP_IRQ interrupt handling procedures to clear the pending bit of the EXTI line 19;
5. Set the MPEN or WFEN (or both) bit in ENET_MAC_WUM register to enable Magic Packet or Remote Wakeup frame (or both) detection;
6. Setting PWD bit in ENET_MAC_WUM register to enter power-down state;
7. Setting REN bit in ENET_MAC_CFG register to make MAC's receive function work;
8. Make MCU enter Deep-sleep mode;
9. After received a wakeup type frame, the Ethernet module exits the power-down state;
10. Reading the ENET_MAC_WUM register to clear the power management event flags. Enable MAC's transmit function and enable TxDMA and RxDMA;
11. Initialize the MCU system clock: enable HXTAL and configure the RCU unit.

### Remote wakeup frame filter register

Wakeup frame filter register is made up of eight different registers but shared the same register offset address. So the inner pointer points the next filter register when the filter register address is accessed by writing or reading. Whatever operation, write or read, it is strongly recommended to operate eight times sequentially. This means continuously write 8 times will configure the filter registers and continuously read 8 times will get the values of filter registers.

**Figure 43-11. Wakeup frame filter register**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Wakeup Frame Filter Register 0 | Filter 0 Byte Mask | | | | | | | |
| Wakeup Frame Filter Register 1 | Filter 1 Byte Mask | | | | | | | |
| Wakeup Frame Filter Register 2 | Filter 2 Byte Mask | | | | | | | |
| Wakeup Frame Filter Register 3 | Filter 3 Byte Mask | | | | | | | |
| Wakeup Frame Filter Register 4 | Reserve | Filter 3 Command | Reserve | Filter 2 Command | Reserve | Filter 1 Command | Reserve | Filter 0 Command |
| Wakeup Frame Filter Register 5 | Filter 3 Offset | | Filter 2 Offset | | Filter 1 Offset | | Filter 0 Offset | |
| Wakeup Frame Filter Register 6 | Filter 1 CRC - 16 | | | | Filter 0 CRC - 16 | | | |
| Wakeup Frame Filter Register 7 | Filter 3 CRC - 16 | | | | Filter 2 CRC - 16 | | | |

- Filter n Byte mask

This register field defines using which bytes of the frame to determine the received frame is wakeup frame or not by filter n (n=0, 1, 2, 3). Bit 31 must be set to 0. Bit 30 to bit 0 are valid byte mask. If bit m (m means byte number) is set, the filter n offset + m of the receiving frame is calculated by the CRC unit, conversely, filter n offset + m is ignored.

- Filter n command

This four bits command controls the operation of the filter n. The bit 3 of the field is address type selection bit. If this bit is 1, the detection only detects a multicast frame and if this bit is 0, the detection only detects a unicast frame. Bit 2 and bit 1 must be set to 0. Bit 0 is the filter switch bit. Setting it to 1 means enable and 0 means disable.

- Filter n offset

It is used in conjunction with filter n byte mask field. This register specifies offset (within the frame) of the first byte which the filter n uses to check. The minimum allowable value is 12, it represents the byte 13 in the frame (offset value 0 indicates the first byte of the frame).

- Filter n CRC-16

This register field contains the filter comparing CRC-16 code which is used for comparing the calculated CRC-16 from frame data.

### 43.3.6. Precision time protocol: PTP

The majority of protocols are implemented by the UDP layer application software. The PTP module of the MAC is mainly to recording the transmitting and receiving PTP packets' precision time and returning it to application.

Specific details about the precise time protocol (PTP) please see the document "IEEE Standard 1588™".

### Reference clock source

System reference time in Ethernet is maintained by a 64-bit register whose high 32-bit indicates ‘second’ time and low 32-bit indicates ‘subsecond’, this is defined in IEEE 1588 specification.

The input PTP reference clock is used to drive the system reference time (also called system time for short) and capture timestamp value for PTP frame. The frequency of this reference clock must be configured no less than the resolution of timestamp counter. The synchronization accuracy between the master node and slave node is around 0.1us.

### Synchronization accuracy

The accuracy of time synchronization depends on the following factors:

- PTP reference clock input period.
- Characteristics of the oscillator (drift).
- Frequency of the synchronization procedure.

### System time calibration

PTP input reference clock is used to update 64-bit PTP system time. The PTP system time is used as the source to record transmission / reception frame’s timestamp. The system time initialization and calibration support two methods: coarse method and fine method. The purpose of calibration is to correct the frequency offset.

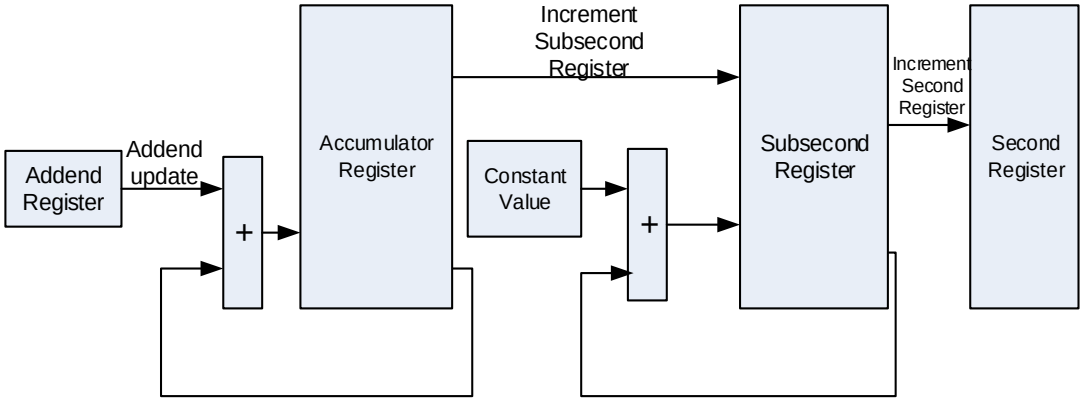
If the coarse correction method is selected, application can configure PTP timestamp update register (ENET_PTP_TSUH and ENET_PTP_TSUL) for system time initialization or correction. If TMSSTI bit is set, PTP timestamp update register is used for initialization and if TMSSTU bit is set, PTP timestamp update register is used for adjust system time by adding or subtracting.

If fine correction method is selected, operation is different. The fine correction method corrects system time not in a single clock cycle. The fine correction frequency can be configured by application to make slave clock frequency smoothly adapt master clock without unpredictability large jitter.

This method is referred to the value of ENET_PTP_TSADDEND added to the accumulator in each HCLK cycle. PTP module will produce a pulse to increase the value of ENET_PTP_TSL register when the accumulator overflowed. The increased value when this pulse occurs is in ENET_PTP_SSINC register.

*Figure 43-12. System time update using the fine correction method* shows the fine correction algorithm process:

**Figure 43-12. System time update using the fine correction method**




The following concrete example is used to descript the fine correction method how to update the system time:

Assuming the accuracy of the system time update circuit required to achieve 25ns, which means the frequency of update is 40MHz. If the reference clock of HCLK is 72MHz, the frequency ratio is calculated as 72 / 40, result is 1.8. Hence, the addend (TMSA bit in ENET_PTP_TSADDEND register) value to be set is $2^{32}$ / 1.8, which is equal to 0x8E38 E38E. If the reference clock frequency drifts lower, for example, down to 68MHz, the frequency ratio changes to 68 / 40 = 1.7, the value to be set in the addend register is $2^{32}$ / 1.7 = 0x9696 9697. If the reference clock drift higher, for example, up to 76MHz, the value should to be set in the ENET_PTP_TSADDEND register is $2^{32}$ / 1.9 = 0x86BC A1AF. Initially, the slave clock frequency is set to Clock Addend Value (0) in the addend register. This value is calculated as above. In addition to configuring the addend counter, application also needs to set subsecond increment register to ensure to achieve the precision of 25ns. The value of the register is to update values of timestamp low 32-bit register after accumulator register overflow. Because the STMSS[30:0] bits in the ENET_PTP_TSL register represents the subsecond value of system time, the precision is $10^9$ns / $2^{31}$=0.46ns. So in order to make the system time accuracy to 25ns, subsecond increment register value should be set to 25 / 0.46 = 0d54.

**Note:** The algorithm described below based on constant delay transferred between master and slave devices (Master-to-Slave-Delay). Synchronous frequency ratio will be confirmed by the algorithm after a few Sync cycles.

Algorithm is as follows:

- Define the master sends a SYNC message to slave time: MSYNCT (n).
  Define the slave local time: SLOCALT (n).
  Define the master local time: MLOCALT (n).
  Calculation: MLOCALT (n) = MSYNCT (n) + Master-to-Slave-Delay (n).

- Define the master clock count number between two SYNC message sent: MCLOCKC(n).
  Calculation: MCLOCKC (n) = MLOCALT (n) – MLOCALT (n-1).
  Define the slave clock count number between two received SYNC messages: SCLOCKC (n).
  Calculation: SCLOCKC (n) = SLOCALT (n) - SLOCALT (n-1).

- Define the difference between these two count numbers: DIFFCC (n).
  Calculation: DIFFCC (n) = MCLOCKC (n) - SCLOCKC (n).
- Define the slave clock frequency-adjusting factor: SCFAF (n).
  Calculation: SCFAF (n) = (MCLOCKC (n) + DIFFCC (n)) / SCLOCKC (n).
- Define the Clock Addend Value for addend register: Clock Addend Value (n).
  Clock Addend Value (n) = SCFAF (n) * Clock Addend Value (n-1).

**Note:** During the actual operation, application may need more than once SYNC message between master and slave to lock.

### System time initialization procedure

Setting TMSEN bit in ENET_PTP_TSCTL register to 1, timestamp function is enabled. Each time after this bit is set from reset, application must initialize the timestamp counter at first. Initialization steps as follow:

1. Setting TMSTIM in the ENET_MAC_INTMSK register to mask the timestamp trigger interrupt;
2. Setting TMSEN in the ENET_PTP_TSCTL register to enable timestamp function;
3. Configure the subsecond increment register according to the PTP clock frequency precision;
4. If application hopes to use fine correction method, configure the timestamp addend register and set TMSARU in the ENET_PTP_TSCTL register to 1. If application hopes to use coarse correction method, please jump directly to step 7 and step 4-6 can be ignored;
5. Poll the TMSARU in the ENET_PTP_TSCTL register until it is cleared;
6. Set TMSFCU in the ENET_PTP_TSCTL register to 1 to choose fine correction method;
7. Configure the timestamp update high and low register with the value of system time application wants to initialize;
8. Send initialization command by setting TMSSTI in the ENET_PTP_TSCTL register;
9. The timestamp counter starts counting as soon as the initialization process complete.

### System time update steps

#### Coarse correction method

1. Program the offset (may be negative) value in the timestamp update high and low registers;
2. Set bit 3 (TMSSTU) in the ENET_PTP_TSCTL register to update the timestamp register;
3. Poll TMSSTU bit until it is cleared.

#### Fine correction method

1. Calculate the value of the desired system clock rate corresponding to the addend register (***System time calibration*** has explained before);
2. Program the addend register, and set the TMSARU in ENET_PTP_TSCTL register;

3. Program the target high and low register and reset the TMSTIM of the ENET_MAC_INTMSK register to allow time stamp interrupt;
4. Set TMSITEN in ENET_PTP_TSCTL register;
5. When an interrupt is generated by this event, read out the value of ENET_MAC_INTF register and clear the corresponding interrupt flag;
6. Rewrite the old value of addend register to timestamp addend register and set TMSARU in ENET_PTP_TSCTL register.

### Transmission and reception of frames with the PTP feature

After enabled the IEEE 1588 (PTP) timestamp function, timestamp is recorded when the frame's SFD field is outputting from the MAC or the MAC receives a frame's SFD field. Each transmitted frame can be marked in TxDMA descriptor to indicate whether a timestamp should be captured or not, which is unrelated with whether the transmitted frame has PTP feature or not, and the timestamp of all received frames will be recorded if ARFSEN bit in ENET_PTP_TSCTL register is set. If ARFSEN is reset, the received frame which passed the address filer should be matched with the configuration in ENET_PTP_TSCTL register. In another word, only the frame matched the PTP configuration is marked a PTP frame, and timestamp will be recorded in descriptor. To be marked as a PTP frame, the received frame PTP version should be coincide with PFSV bit and then the corresponding frame type enable bit (bit 13 to bit 11 in register ENET_PTP_TSCTL) is set. Specially, the non-IP payload PTP frame (PTP on normal 802.3 Ethernet frame), also the DA should be the special MAC address (e.g. the DA should be 0x0e00 00c2 8001 for PDELAY_REQ / PDELAY_RESP / PDELAY_RESP_FOLLOW_UP message type, and the DA address 0x0000 0019 1B01 for other message type, detailed informations refer to Specification IEEE1588-2008). If MAFEN is set, this special MAC address can be extended to MAC address1-3 with SAF is reset.

Together with the state information of frame, the recorded timestamp value will also be stored in the corresponding transmission / reception descriptor. The 64-bit timestamp information of transmission frame is written back to the transmit descriptor and the 64-bit timestamp information of reception frame is written back to the receive descriptor. See the detailed description in "***Transmit DMA descriptor with IEEE 1588 timestamp format***" and "***Receive DMA descriptor with IEEE 1588 timestamp format***".

### Internal connection trigger

MAC can provide trigger interrupt when the system time is no less than the expected time. Using an interrupt imports a known latency and an uncertainty in the command execution time. Set the TSCFG5[4:0]/TSCFG4[4:0]/TSCFG3[4:0] bit-field in SYSCFG_TIMERxCFG0(x = 1/2/30/31) register to 5'b01001 can make ENET0 signal internally connected to the ITI4 input of event mode configuration/pause mode configuration/restart mode configuration in TIMERx(x = 1/2/30/31). And set the TSCFG7[4:0]/TSCFG6[4:0] bit-field in SYSCFG_TIMERxCFG1(x = 1/2/30/31) register to 5'b01001 can make ENET1 signal internally connected to the ITI4 input of restart + event mode configuration/external clock mode 0 configuration in TIMERx(x = 1/2/30/31).. For this feature designed, no uncertainty is introduced because the clock of the TIMER1/2/30/31 and PTP reference clock (HCLK) are

synchronous.

**PPS output signal**

Enable the ENET0 and ENET1 PPS output function by configuring ETH0_PPS_OUT pin to AF11 and ETH0_PPS_OUT pin to AF6, respectively. This function can output a signal with the pulse width of 125ms by default (other width is detailed in ***PTP PPS control register (ENET_PTP_PPSCTL)***) which can be used to check the synchronization between all nodes in the network. To test the difference between the slave clock and the master clock, both of the slave and master can output PPS(pulse-per-second) and connect them to one oscilloscope for clock measurement.

### 43.3.7. Example for a typical configuration flow of Ethernet

After power-on reset or system reset, the following operation flow is a typical process for application to configure and run Ethernet:

- **Enable Ethernet clock.**

  Program the RCU module to enable the HCLK and Ethernet Tx / Rx clock.

- **Setup the communication interface.**

  Configure SYSCFG_PMCFG to define which interface mode is selected (MII or RMII). Configure GPIO module to make selected PADs to alternate function.

- **Wait the resetting complete**

  Polling the ENET_DMA_BCTL register until the SWR bit is reset. (SWR bit is set by default after power-on reset or system reset).

- **Obtain and configure the parameters in PHY register**

  According to the frequency of HCLK, configure the SMI clock frequency and access external PHY register to obtain the information of PHY (e.g. support Half / Full duplex or not, support 10M / 100Mbit speed or not, and so on). Based on supported mode of external PHY, configure ENET_MAC_CFG register consistent with PHY register.

- **Initialize the DMA in Ethernet module for transaction**

  Configure the ENET_DMA_BCTL, ENET_DMA_RDTADDR, ENET_DMA_TDTADDR, ENET_DMA_CTL registers to initialize the DMA module. (Detailed information refer to ***DMA controller description***).

- **Initialize the physical memory space for descriptor table and data buffer**

  According to the address value in ENET_DMA_RDTADDR and ENET_DMA_TDTADDR register, program transmitting and receiving descriptors (with DAV=1) and data buffer.

- **Enable MAC and DMA module to start transmit and receive**

  Set TEN and REN bit in ENET_MAC_CFG register to make MAC work for transmit and

receive. Set STE and SRE bit in ENET_DMA_CTL register to make DMA controller work for transmit and receive.

- **If transmitting frames is needed**

1. Choose one or more programmed transmitting descriptor, write the transmit frame data into buffer address which is decided in Transmit Descriptor;
2. Set the DAV bit in these one or more transmit frame descriptor;
3. Write any value in ENET_DMA_TPEN register to make TxDMA exit suspend state and start transmitting;
4. There are two methods for application to confirm whether current transmitting frame is complete or not. The first method is that application can poll the DAV bit of current transmit descriptor until it is reset, this means the transmitting is complete. The second method can be used only when INTC=1. Application can poll the TS bit in ENET_DMA_STAT register until it is set, this means the transmitting is complete.

- **If receiving frames is enabled**

1. Check the first receive descriptor in descriptor table (whose address is configured in ENET_DMA_RDTADDR register);
2. If DAV bit in Receive Descriptor0 is reset, then the descriptor is used and receive buffer space has stored the receive frame;
3. Handling this receive frame data;
4. Set DAV bit of this descriptor to release this descriptor for new frame receiving;
5. Check next descriptor in table, then goes to Step 2.

### 43.3.8. Ethernet interrupts

There are two interrupt vectors in Ethernet module. The first interrupt vector is made up of normal operation interrupts and the second vector is made up of WUM events for wakeup which is mapped to the EXTI line 19.

All of the MAC and DMA controller interrupt are connected to the first interrupt vector. The description for the MAC interrupt and DMA controller interrupt are showed behind.

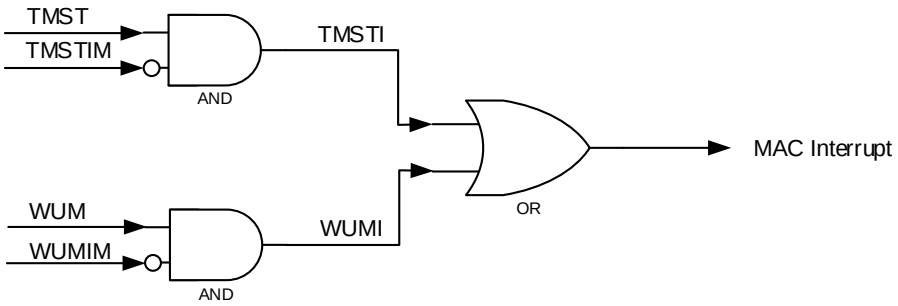
The WUM block event is connected to the second interrupt vector. The event can be remote wakeup frame received event or / and Magic Packet wakeup frame received event. This interrupt is inner mapped on the EXTI line 19. So, if the EXTI line 19 is enabled and configured to trigger by rising edge, the Ethernet WUM event can make the system exiting Deep-sleep mode after a WUM event occurred. In addition, if the WUM interrupt is not masked, both the EXTI line 19 interrupt and Ethernet normal interrupt to CPU are both generated.

**Note:** Because of the WUM registers are designed in RX_CLK domain, clear these registers by reading them will need a long time delay (depends on the frequency disparity between HCLK and RX_CLK). To avoid entering the same event interrupt twice, it's strongly recommended that application polls the WUFR and MPKR bit until they reset to zero during the interrupt service routine.

### MAC interrupts

All of the MAC events can be read from ENET_MAC_INTF and each of them has a mask bit for masking corresponding interrupt. The MAC interrupt is logical ORed of all interrupts.

**Figure 43-13. MAC interrupt scheme**



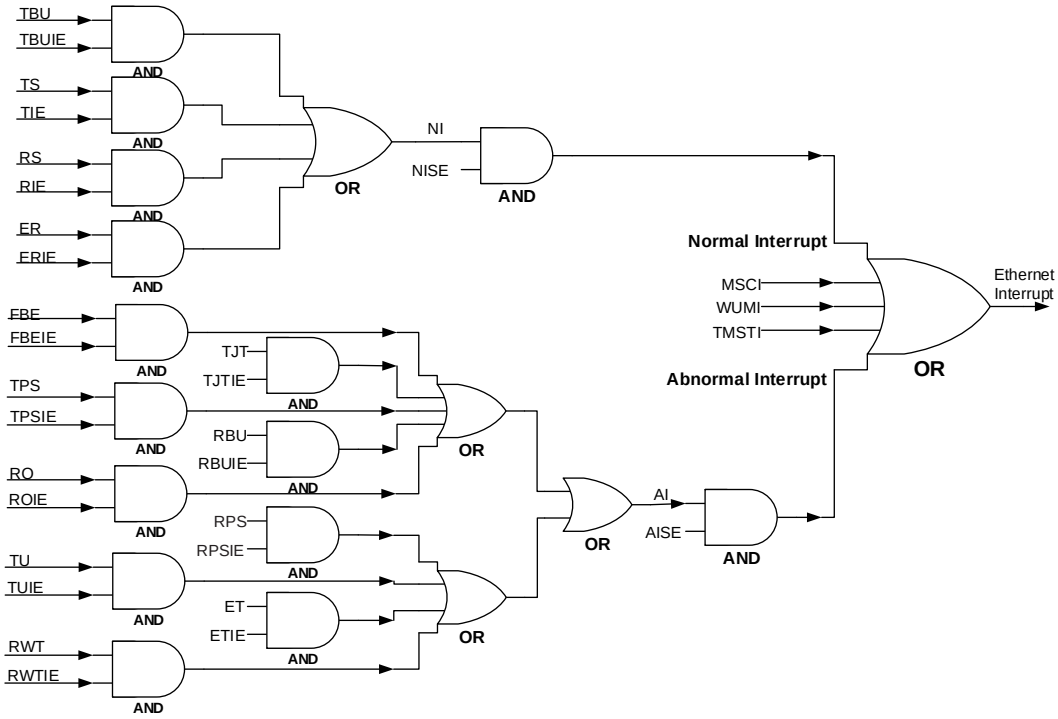

### DMA controller interrupts

The DMA controller has two types of event: Normal and Abnormal.

No matter what type the event is, it has an enable bit (just like mask bit) to control the generating interrupt or not. Each event can be cleared by writing 1 to it. When all of the events are cleared or all of the event enable bits are cleared, the corresponding summary interrupt bit is cleared. If both normal and abnormal interrupts are cleared, the DMA interrupt will be cleared.

*Figure 43-14. Ethernet interrupt scheme* shows the Ethernet module interrupt connection:

**Figure 43-14. Ethernet interrupt scheme**


TBU
TBUIE
AND
TS
TIE
AND
RS
RIE
AND
ER
ERIE
AND
OR
NI
NISE
AND
Normal Interrupt
MSCI
WUMI
TMSTI
Ethernet Interrupt
OR
Abnormal Interrupt
FBE
FBEIE
AND
TJT
TJTIE
AND
TPS
TPSIE
AND
RBU
RBUIE
AND
OR
RO
ROIE
AND
RPS
RPSIE
AND
OR
AI
AISE
AND
TU
TUIE
AND
ET
ETIE
AND
OR
RWT
RWTIE
AND

## 43.4. Register definition

ENET0 base address: 0x4002 8000
ENET1 base address: 0x4002 A000

### 43.4.1. MAC configuration register (ENET_MAC_CFG)

Address offset: 0x0000
Reset value: 0x0000 8000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).
This register configures the operation mode of the MAC. It also configures the MAC receiver and MAC transmitter operating mode.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | TFCD | Reserved | WDD | JBD | Reserved | | IGBS[2:0] | | | CSD |
| | | | | | | rw | | rw | rw | | | | rw | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SPD | ROD | LBM | DPM | IPFCO | RTD | Reserved | APCD | BOL[1:0] | | DFC | TEN | REN | Reserved | |
| | rw | rw | rw | rw | rw | rw | | rw | rw | | rw | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25 | TFCD | Type Frame CRC Dropping<br>0: FCS field (last 4 bytes) of frame will not be dropped before forwarding<br>1: FCS field (last 4 bytes) of frame will be dropped before forwarding<br>**Note:** This bit only valid when LT field of frame greater than 0x0600. |
| 24 | Reserved | Must be kept at reset value. |
| 23 | WDD | Watchdog disable bit<br>This bit indicates the maximum bytes for receiving, data beyond this will be cut off.<br>0: The received frame that Less than or equals to 2048 bytes is allowed by MAC<br>1: The watchdog timer that on the receiver is disabled by MAC. And the received frame up to 16384 bytes is allowed by MAC. |
| 22 | JBD | Jabber disable bit<br>This bit indicates the maximum bytes for transmitting data, data beyond this will be cut off.<br>0: The maximum transmission byte is 2048<br>1: The maximum transmission byte can be 16384 |
| 21:20 | Reserved | Must be kept at reset value. |
| 19:17 | IGBS[2:0] | Inter frame gap bit selection bits<br>These bits can select the minimum inter frame gap bit time between two neighboring frames during transmission. |

| | | |
|---|---|---|
| | | 0x0: 96 bit times<br>0x1: 88 bit times<br>0x2: 80 bit times<br>0x3: 72 bit times<br>0x4: 64 bit times<br>0x5: 56 bit times (For Half-duplex, must be reserved)<br>0x6: 48 bit times (For Half-duplex, must be reserved)<br>0x7: 40 bit times (For Half-duplex, must be reserved) |
| 16 | CSD | Carrier sense disable bit<br>0: The carrier sense error is generated by MAC transmitter, and the transmission will be aborted.<br>1: The MII CRS signal is ignored by MAC transmitter while in frame transmitting. Loss of carrier error and no carrier error will not be generated. |
| 15 | Reserved | Must be kept at reset value. |
| 14 | SPD | Fast Ethernet speed bit<br>Indicates the speed in Fast Ethernet mode:<br>0: 10 Mbit / s<br>1: 100 Mbit / s |
| 13 | ROD | Receive own disable bit<br>When in Full-duplex mode, this bit can be ignored.<br>0: The packets that transmitting from PHY are all received by MAC<br>1: Receiving frames from PHY is disabled by MAC |
| 12 | LBM | Loopback mode bit<br>0: The MAC is configured in normal mode<br>1: The MAC is configured in loopback mode at the MII |
| 11 | DPM | Duplex mode bit<br>0: Half-duplex mode enable<br>1: Full-duplex mode enable |
| 10 | IPFCO | IP frame checksum offload bit<br>0: The checksum offload function in the receiver is disabled<br>1: IP frame checksum offload function enabled for received IP frame |
| 9 | RTD | Retry disable bit<br>When in Full-duplex mode, this bit can be ignored.<br>0: Up to 16 times retries based on the settings of BOL is attempted by MAC<br>1: Only 1 transmission is attempted by MAC |
| 8 | Reserved | Must be kept at reset value. |
| 7 | APCD | Automatic pad / CRC drop bit<br>This bit only valid for a non tagged frame and its length field value is equal or less than 1536. |

| | | |
|---|---|---|
| | | 0: The MAC forwards all received frames without modify it<br>1: The MAC strips the Pad / FCS field on received frames |
| 6:5 | BOL[1:0] | Back-off limit bits<br>When in Full-duplex mode, these bits can be ignored.When a collision occurred, the MAC needs to retry sending current frame after delay some time. The base time unit for this delay time (dt) called slot time which means 1 slot time is equal to 512 bit times. This delay time (dt) is a random integer number calculated by following formula: $0 \leq dt < 2^{k}$.<br>0x0: k = min (n, 10)<br>0x1: k = min (n, 8)<br>0x2: k = min (n, 4)<br>0x3: k = min (n, 1),<br>n = number of times for retransmission attempt |
| 4 | DFC | Deferral check bit<br>When in Full-duplex mode, this bit can be ignored.<br>0: Disable the deferral check function of MAC. Until the CRS signals changed to inactive, the MAC defers sending.<br>1: Enable the deferral check function of MAC. If deferred more than 24288 bit times, excessive deferral error occurs and MAC abort transmitting frame. If CRS signal active during deferral time running, the deferral time will reset and restart. |
| 3 | TEN | Transmitter enable bit<br>0: The MAC transmit function is disabled after finish the transmission of the current frame, and no frames to be transmitted anymore.<br>1: The transmit function of the MAC is enabled for transmission |
| 2 | REN | Receiver enable bit<br>0: The MAC reception function is disabled after finish the reception of the current frame, and no frames will be received anymore.<br>1: The MAC reception function is enabled for receiving frames |
| 1:0 | Reserved | Must be kept at reset value. |

## 43.4.2. MAC frame filter register (ENET_MAC_FRMF)

Address offset: 0x0004
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the filtering method for receiving frames

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FAR | Reserved | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | HPFLT | SAFLT | SAIFLT | PCFRM[1:0] | | BFRMD | MFD | DAIFLT | HMF | HUF | PM |

rw rw rw rw rw rw rw rw rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | FAR | Frames all received bit<br>This bit controls the receive filter function.<br>0: Only the frame passed the filter can be forwarded to application<br>1: All received frame are forwarded to application. But filter result will also be updated to receive descriptor status. |
| 30:11 | Reserved | Must be kept at reset value. |
| 10 | HPFLT | Hash or perfect filter bit<br>0: If the HUF or HMF bit is set, only frames that match the hash filter are passed.<br>1: If the HUF or HMF bit is set, the receive filter passes frames that match either the perfect filtering or the hash filtering. |
| 9 | SAFLT | Source address filter bit<br>Enable source address filtering function besides destination address filtering.<br>The filter also compares the SA field value in received frames with the values configured in the enabled SA registers. If SA comparison matches, the SA match bit in the receive descriptor status is set high.<br>0: Source address function in filter disable<br>1: Source address function in filter enable |
| 8 | SAIFLT | Source address inverse filtering bit<br>This bit makes the result of SA matching inverse.<br>0: Not inverse for source address filtering<br>1: Inverse source address filtering result. When SA matches the enabled SA registers, filter marks it as failing the SA address filter. |
| 7:6 | PCFRM[1:0] | Pass control frames bits<br>These bits set the forwarding conditions for all control frames (including unicast and multicast pause frame).<br>For pause control frame, the processing (not forwarding) depends only on RFCEN in ENET_MAC_FCTL[2].<br>0x0: The MAC does not forward any control frames to the application<br>0x1: The MAC forwards any control frames except pause control frames to the application<br>0x2: Even if the control frames failed the address filter, the MAC forwards all of them to application<br>0x3: Only the control frames pass the address filter, the MAC forwards them to application |
| 5 | BFRMD | Broadcast frames disable bit<br>0: Ignore the address filters, and all received broadcast frames is passed.<br>1: All received broadcast frames is filtered by address filters |
| 4 | MFD | Multicast filter disable bit |

| | | |
|---|---|---|
| | | 0: Multicast filter is enabled. The filtering mode of multicast frame is determined by HMF bit.<br>1: Multicast filter is disabled. All received multicast frames are passed. The first bit in the destination address field of multicast frames is '1', but not all bits in the destination are '1'. |
| 3 | DAIFLT | Destination address inverse filtering bit<br>This bit makes the result of DA filtering inverse.<br>0: Not inverse DA filtering result<br>1: Inverse DA filtering result |
| 2 | HMF | Hash multicast filter bit<br>0: The filter uses perfect mode for filtering multicast frame.<br>1: The filter uses hash mode for filtering multicast frame |
| 1 | HUF | Hash unicast filter bit<br>0: The filter uses perfect mode for filtering unicast frame<br>1: The filter uses hash mode for filtering unicast frame |
| 0 | PM | Promiscuous mode bit<br>This bit can make the filter bypassed which means all received frames are thought pass the filer and DA / SA filtering result status in descriptor is always '0'.<br>0: Promiscuous mode disabled<br>1: Promiscuous mode enabled |

### 43.4.3. MAC hash list high register (ENET_MAC_HLH)

Address offset: 0x0008
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HLH[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HLH[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | HLH[31:0] | Hash list high bits<br>These bits take the high 32-bit value of hash list. |

### 43.4.4. MAC hash list low register (ENET_MAC_HLL)

Address offset: 0x000C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HLL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HLL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | HLL[31:0] | Hash list low bits<br>These bits take the low 32-bit value of hash list. |

## 43.4.5. MAC PHY control register (ENET_MAC_PHY_CTL)

Address offset: 0x0010

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PA[4:0] | | | | | PR[4:0] | | | | | Reserved | CLR[2:0] | | | PW | PB |
| rw | | | | | rw | | | | | | rw | | | rw | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:11 | PA[4:0] | PHY address bits<br>These bits choose which PHY device is to be accessed. |
| 10:6 | PR[4:0] | PHY register bits<br>These bits choose the register address in selected PHY device. |
| 5 | Reserved | Must be kept at reset value. |
| 4:2 | CLR[2:0] | Clock range bits<br>MDC clock divided factor select which is decided by HCLK frequency range.<br>0x0: HCLK/42 (HCLK range: 60-100 MHz)<br>0x1: HCLK/62 (HCLK range: 100-150 MHz)<br>0x2: HCLK/16 (HCLK range: 20-35 MHz)<br>0x3: HCLK/26 (HCLK range: 35-60 MHz)<br>0x4: HCLK/102 (HCLK range: 150-250 MHz)<br>0x5: HCLK/124 (HCLK range: 250-300 MHz)<br>0x6: HCLK/142 (HCLK range: 300-350 MHz) |

other: Reserved

| | | |
|---|---|---|
| 1 | PW | PHY write bit<br>This bit indicates the PHY operation mode.<br>0: Sending read operation to PHY<br>1: Sending write operation to PHY |
| 0 | PB | PHY busy bit<br>This bit indicates the running state of operation on PHY. Application sets this bit to 1 and should wait it cleared by hardware. Application must make sure this bit is zero before writing data to ENET_MAC_PHY_CTL register and reading / writing data from / to ENET_MAC_PHY_DATA register. |

### 43.4.6. MAC PHY data register (ENET_MAC_PHY_DATA)

Address offset: 0x0014
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | PD[15:0] | PHY data bits<br>For reading operation, these bits contain the data from external PHY. For writing operation, these bits contain the data will be sent to external PHY. |

### 43.4.7. MAC flow control register (ENET_MAC_FCTL)

Address offset: 0x0018
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the generation and reception of the control frames.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PTM[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Reserved | DZQP | Reserved | PLTS[1:0] | UPFDT | RFCEN | TFCEN | FLCB / BKPA |
|---|---|---|---|---|---|---|---|
| | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | PTM[15:0] | Pause time bits<br>These bits configured the pause time filed value in transmit pause control frame. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | DZQP | Disable Zero-quanta pause bit<br>0: Enable automatic zero-quanta generation function for pause control frame<br>1: Disable the automatic zero-quanta generation function for pause control frame |
| 6 | Reserved | Must be kept at reset value. |
| 5:4 | PLTS[1:0] | Pause low threshold bits<br>These bits configure the threshold of the pause timer for retransmitting frames automatically. Application must make sure the low threshold bits are greater than 0 and less than configured pause time. The low threshold calculation formula is PTM-PLTS. For example, if PTM = 0x80 (128 slot-times), and PLTS = 0x1 (28 slot-times), then the second pause frame is automatically transmitted when pause timer counted at 100 (128 - 28) slot-times after the first pause frame is transmitted.<br>0x0: 4 slot times is subtracted by pause time<br>0x1: 28 slot times is subtracted by pause time<br>0x2: 144 slot times is subtracted by pause time<br>0x3: 256 slot times is subtracted by pause time<br>**Note:** One slot time equals the time of transmitting 512 bits on the MII interface. |
| 3 | UPFDT | Unicast pause frame detect bit<br>0: Only the unique multicast address for pause frame which is specified in IEEE802.3 can be detected<br>1: Besides the unique multicast address, MAC can also use the MAC0 address (ENET_MAC_ADDR0H and ENET_MAC_ADDR0L register) to detecting pause frame. |
| 2 | RFCEN | Receive flow control enable bit<br>0: Decode function for pause frame is disabled<br>1: Enable decoding function for the received pause frame and process it. The MAC disables its transmitter for a specified (pause time field value in received frame) time. |
| 1 | TFCEN | Transmit flow control enable bit<br>0: Disable the flow control operation in the MAC. Both pause frame sending in Full-duplex mode and back-pressure feature in Half-duplex mode are not performed.<br>1: Enable the flow control operation in the MAC. Both pause frame sending in Full-duplex mode and back-pressure feature in Half-duplex mode can be performed by transmitter. |

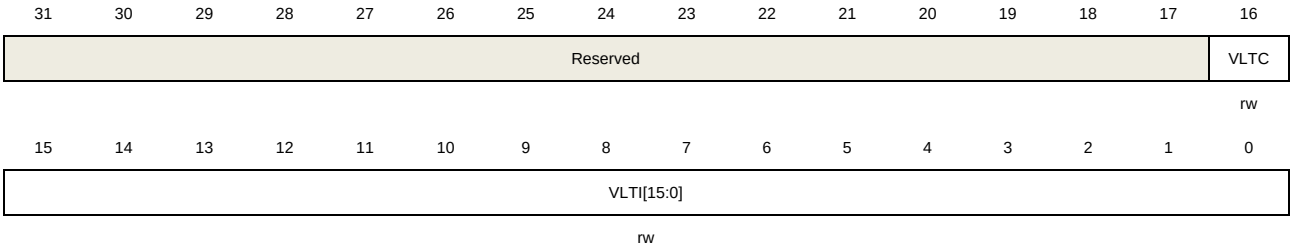
| | | |
|---|---|---|
| 0 | FLCB / BKPA | Flow control busy / back pressure activate bit<br>This bit only valid when TFCEN is set.<br>This bit can send a pause frame in Full-duplex mode or activate the back pressure function in Half-duplex mode by application.<br>For Full-duplex mode, application must make sure this bit is 0 before writing ENET_MAC_FCTL register. After set by application, MAC sends a pause frame to interface and this bit will keep set until the pause frame has completed transmitting.<br>For Half-duplex mode, MAC can enter back-pressure state by application setting this bit. When the MAC is in back-pressure state, any frame presented on interface will make the MAC send a JAM pattern to inform outside a collision occurred. |

### 43.4.8. MAC VLAN tag register (ENET_MAC_VLT)

Address offset: 0x001C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th byte (length / type field) of the receiving frame with 0x8100, and the following 2 bytes (the 15th and 16th byte) are compared with the VLAN tag.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | VLTC | 12-bit VLAN tag comparison bit<br>This bit selects 12 or 16 bit VLAN tag for comparison.<br>0: All 16 bits (the 15th and 16th byte) of the VLAN tag in received frame are used for comparison<br>1: Only low 12 bits of the VLAN tag in received frame are used for comparison |
| 15:0 | VLTI[15:0] | VLAN tag identifier (for receive frames) bits<br>These bits are configured for detecting VLAN frame using 802.1Q VLAN tag format. The format shows below:<br>VLTI[15:13]: UP (user priority)<br>VLTI[12]: CFI (canonical format indicator)<br>VLTI[11:0]: VID (VLAN identifier)<br>When comparison bits (VLTI[11:0] if VLTC=1 or VLTI[15:0] if VLTC=0) are all zeros, VLAN tag comparison is bypassed and every frame with type filed value of 0x8100 is considered a VLAN frame. |

When comparison bits not all zeros, VLAN tag comparison use bit VLTI[11:0] (if VLTC=1) or VLTI[15:0] (if VLTC=0) for checking.

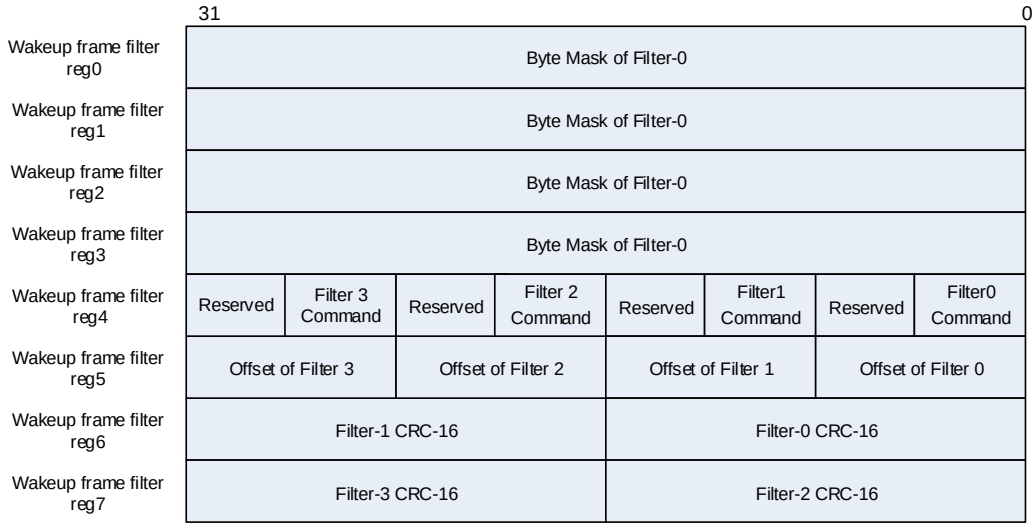
### 43.4.9. MAC remote wakeup frame filter register (ENET_MAC_RWFF)

Address offset: 0x0028
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

The MAC remote wakeup frame filter register is actually a pointer to eight (with same address offset) such wakeup frame filter registers. Eight sequential write operations to this address with the offset (0x0028) will write all wakeup frame filter registers. Eight sequential read operations from this address with the offset (0x0028) will read all wakeup frame filter registers.

**Figure 43-15. Wakeup frame filter register**

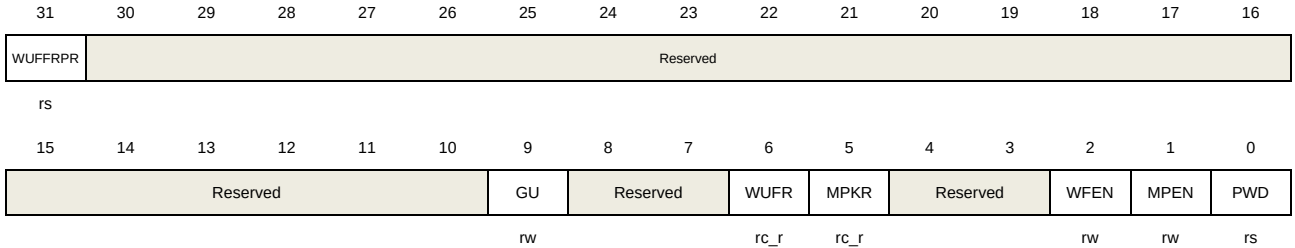



### 43.4.10. MAC wakeup management register (ENET_MAC_WUM)

Address offset: 0x002C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the request of wakeup events and monitors the wakeup events.




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31 | WUFFRPR | Wakeup frame filter register pointer reset bit<br>This bit can reset the inner pointer of ENET_MAC_RWFF register by application set it to 1. Hardware clears it when resetting completes.<br>0: No effect<br>1: Reset the ENET_MAC_RWFF register inner pointer |
| 30:10 | Reserved | Must be kept at reset value. |
| 9 | GU | Global unicast bit<br>0: Not all of received unicast frame is considered to be a wakeup frame<br>1: Any received unicast frame passed address filtering is considered to be a wakeup frame |
| 8:7 | Reserved | Must be kept at reset value. |
| 6 | WUFR | Wakeup frame received bit<br>This bit is cleared when this register is read.<br>0: Has not received the wake-up frame<br>1:The wakeup event was generated due to reception of a wakeup frame |
| 5 | MPKR | Magic packet received bit<br>This bit is cleared when this register is read.<br>0: Has not received the Magic Packet frame<br>1: Received the Magic Packet frame, and generating the wakeup event |
| 4:3 | Reserved | Must be kept at reset value. |
| 2 | WFEN | Wakeup frame enable bit<br>0: Disable generating a wakeup event due to wakeup frame reception<br>1: Enable generating a wakeup event due to wakeup frame reception |
| 1 | MPEN | Magic Packet enable bit<br>0: Disable generating a wakeup event due to Magic Packet reception<br>1: Enable generating a wakeup event due to Magic Packet reception |
| 0 | PWD | Power down bit<br>This bit is set by application and reset by hardware. When this bit is set, MAC drops all received frames. When power-down mode exit because of wakeup event occurred, hardware resets this bit. |

### 43.4.11. MAC debug register (ENET_MAC_DBG)

Address offset: 0x0034

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | TXFF | TXFNE | Reserved | TXFW | TXFRS[1:0] | | PCS | SOMT[1:0] | | MTNI |
| | | | | | | ro | ro | | ro | ro | | ro | ro | | ro |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | RXFS[1:0] | | Reserved | RXFRS[1:0] | | RXFW | Reserved | RXAFS[1:0] | | MRNI |
| | | | | | | ro | | | ro | | ro | | ro | | ro |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25 | TXFF | TxFIFO Full flag<br>0: TxFIFO is not full<br>1: TxFIFO is full |
| 24 | TXFNE | TxFIFO not empty flag<br>0: TxFIFO is empty<br>1: TxFIFO is not empty |
| 23 | Reserved | Must be kept at reset value. |
| 22 | TXFW | TxFIFO is writing<br>0: TxFIFO is not doing write operation<br>1: TxFIFO is doing write operation |
| 21:20 | TXFRS[1:0] | TxFIFO read operation status<br>0x0: TxFIFO read controller is in idle state<br>0x1: TxFIFO read controller is in reading state<br>0x2: TxFIFO read controller is in waiting feedback Tx status from MAC transmitter<br>0x3: TxFIFO read controller is in writing the Tx descriptor status or flushing the TxFIFO |
| 19 | PCS | Pause condition status<br>0: MAC transmitter is not in pause condition<br>1: MAC transmitter is under pause condition and will delay transmitting frame |
| 18:17 | SOMT[1:0] | Status of MAC transmitter<br>0x0: The MAC transmitter controller is in idle state<br>0x1: The MAC transmitter controller is in Waiting feedback of previous frame status or the end of IFG / BACKOFF period<br>0x2: For Full-duplex mode, indicates pause control frame is transmitting<br>0x3: The MAC transmitter controller is in Reading input frame from FIFO for transmission |
| 16 | MTNI | MAC transmit state not idle<br>0: MAC transmitter is in idle state<br>1: MAC transmitter is not in idle state |
| 15:10 | Reserved | Must be kept at reset value. |
| 9:8 | RXFS | RxFIFO state<br>0x0: The RxFIFO is empty<br>0x1: The flow-control low threshold is greater than RxFIFO number of value |

| | | |
|---|---|---|
| | | 0x2: The flow-control high threshold is lower than RxFIFO number of value<br>0x3: The RxFIFO is full |
| 7 | Reserved | Must be kept at reset value. |
| 6:5 | RXFRS[1:0] | RxFIFO read operation status<br>0x0: RxFIFO read controller is in idle state<br>0x1: RxFIFO read controller is in reading state<br>0x2: RxFIFO read controller is reading frame status (including time-stamp)<br>0x3: RxFIFO read controller is flushing frame |
| 4 | RXFW | RxFIFO is writing<br>0: RxFIFO is not doing write operation<br>1: RxFIFO is doing write operation |
| 3 | Reserved | Must be kept at reset value. |
| 2:1 | RXAFS[1:0] | Rx asynchronous FIFO status<br>RXAFS[1]:Rx asynchronous FIFO reading state in HCLK Clock domain<br>RXAFS[0]:Rx asynchronous FIFO writing state in MAC RX_CLK Clock domain |
| 0 | MRNI | MAC receive state not idle<br>0: MAC receiver is in idle state<br>1: MAC receiver is not in idle state |

## 43.4.12. MAC interrupt flag register (ENET_MAC_INTF)

Address offset: 0x0038
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | TMST | Reserved | | MSCT | MSCR | MSC | WUM | Reserved | | |
| | | | | | | rc_r | | | r | r | r | r | | | |

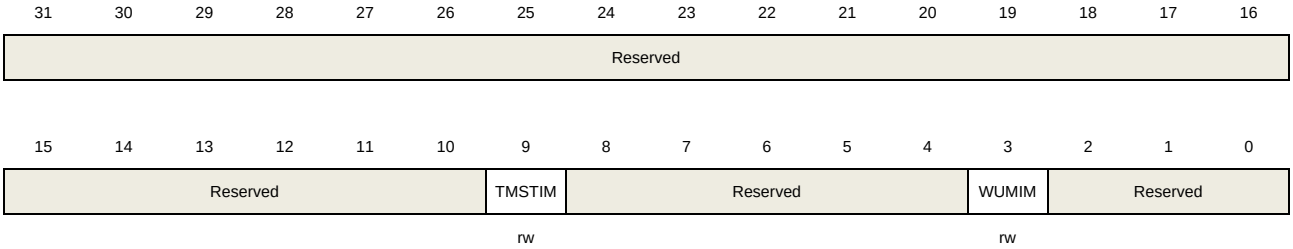
| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | TMST | Time stamp trigger status bit<br>This bit is cleared when ENET_PTP_TSF register is read.<br>0: The system time value is less than the value specified in the the ENET_PTP_ETH and ENET_PTP_ETL registers<br>1: The system time value is no less than the value specified in the ENET_PTP_ETH and ENET_PTP_ETL registers |

| | | |
|---|---|---|
| 8:7 | Reserved | Must be kept at reset value. |
| 6 | MSCT | MSC transmit status bit<br>0: All the bits in register ENET_MSC_TINTF are cleared<br>1: An interrupt is generated in the ENET_MSC_TINTF register |
| 5 | MSCR | MSC receive status bit<br>0: All the bits in register ENET_MSC_RINTF are cleared<br>1: An interrupt is generated in the ENET_MSC_RINTF register |
| 4 | MSC | MSC status bit<br>This bit is logic ORed from MSCT and MSCR bit.<br>0: Both MSCT and MSCR bits in this register are low<br>1: Any of bit 6 (MSCT) or bit 5 (MSCR) is set high |
| 3 | WUM | WUM status bit<br>This bit is logic ORed from WUFR and MPKR bit in ENET_MAC_WUM register.<br>0: Wakeup frame or Magic Packet frame is not received<br>1: A Magic packet or remote wakeup frame is received in power down Mode |
| 2:0 | Reserved | Must be kept at reset value. |

### 43.4.13. MAC interrupt mask register (ENET_MAC_INTMSK)

Address offset: 0x003C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | TMSTIM | Timestamp trigger interrupt mask bit<br>0: Unmask the timestamp interrupt generation<br>1: Mask the timestamp interrupt generation |
| 8:4 | Reserved | Must be kept at reset value. |
| 3 | WUMIM | WUM interrupt mask bit<br>0: Unmask the interrupt generation due to the WUM bit in ENET_MAC_INTF register<br>1: Mask the interrupt generation due to the WUM bit in ENET_MAC_INTF register |

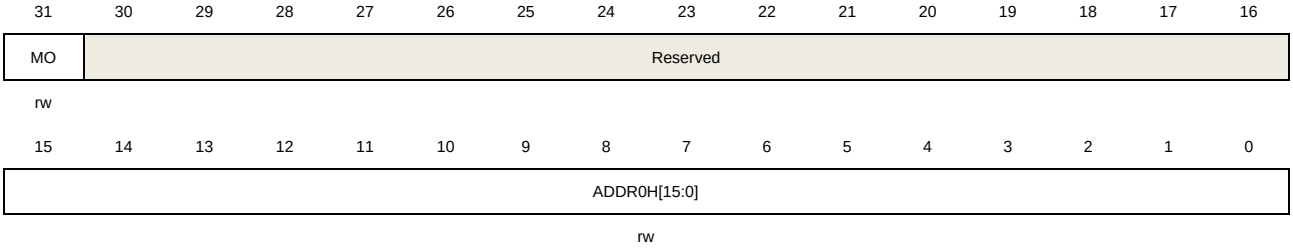
| | | |
|---|---|---|
| 2:0 | Reserved | Must be kept at reset value. |

### 43.4.14. MAC address 0 high register (ENET_MAC_ADDR0H)

Address offset: 0x0040

Reset value: 0x8000 FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



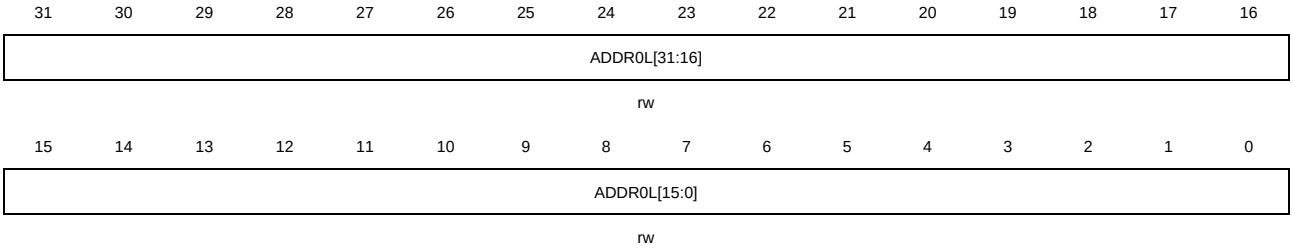

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | MO | Always read 1 and must be kept. |
| 30:16 | Reserved | Must be kept at reset value. |
| 15:0 | ADDR0H[15:0] | MAC address0 high16-bit<br>These bits contain the high 16-bit (bit 47 to 32) of the 6-byte MAC address0. These bits are used for address filtering in frame reception and address inserting in pause frame transmitting during transmit flow control. |

### 43.4.15. MAC address 0 low register (ENET_MAC_ADDR0L)

Address offset: 0x0044

Reset value: 0xFFFF FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR0L[31:0] | MAC addresss0 low 32-bit<br>These bits contain the low 32-bit (bit 31 to 0) of the 6-byte MAC address0. These bits are used for address filtering in frame reception and address inserting in pause frame transmitting during transmit flow control. |

### 43.4.16. MAC address 1 high register (ENET_MAC_ADDR1H)

Address offset: 0x0048

Reset value: 0x0000 FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AFE | SAF | MB[5:0] | | | | | | Reserved | | | | | | | |
| rw | rw | rw | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR1H[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | AFE | Address filter enable bit<br>0: MAC address1 is ignored by address filter for filtering<br>1: MAC address1 is used by address filter for perfect filtering |
| 30 | SAF | Source address filter bit<br>0: Comparing MAC address1 with the destination address field of the received frame<br>1: Comparing MAC address1 with the source address field of the received frame |
| 29:24 | MB[5:0] | Mask byte bits<br>If these bits is set, the destination address / source address corresponding byte of the received frame is not compared with MAC address1. Each bit controls one byte mask as follows:<br>MB[5]: ENET_MAC_ADDR1H[15:8]<br>MB[4]: ENET_MAC_ADDR1H[7:0]<br>MB[3]: ENET_MAC_ADDR1L[31:24]<br>MB[2]: ENET_MAC_ADDR1L[23:16]<br>MB[1]: ENET_MAC_ADDR1L[15:8]<br>MB[0]: ENET_MAC_ADDR1L[7:0] |
| 23:16 | Reserved | Must be kept at reset value. |
| 15:0 | ADDR1H[15:0] | MAC address1 high[47:32] bits<br>This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address1. |

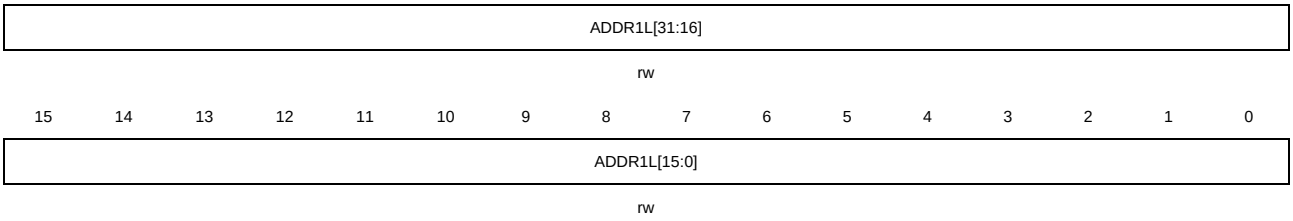
### 43.4.17. MAC address 1 low register (ENET_MAC_ADDR1L)

Address offset: 0x004C

Reset value: 0xFFFF FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



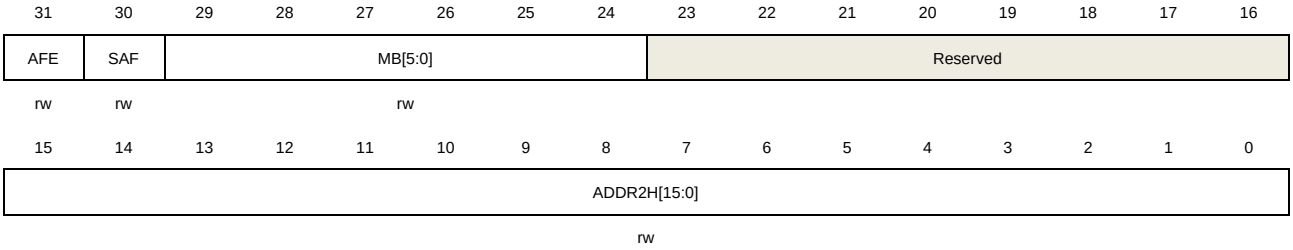

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR1L[31:0] | MAC address1 low 32-bit<br>This field contains the low 32-bit of the 6-byte MAC address1. |

## 43.4.18. MAC address 2 high register (ENET_MAC_ADDR2H)

Address offset: 0x0050

Reset value: 0x0000 FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | AFE | Address filter enable bit<br>0: MAC address2 is ignored by address filter for filtering<br>1: MAC address2 is used by address filter for perfect filtering |
| 30 | SAF | Source address filter bit<br>0: Comparing MAC address2 with the destination address field of the received frame<br>1: Comparing MAC address2 with the source address field of the received frame |
| 29:24 | MB[5:0] | Mask byte bits<br>If these bits is set, the destination address / source address corresponding byte of the received frame is not compared with MAC address2. Each bit controls one byte mask as follows:<br>MB[5]: ENET_MAC_ADDR2H[15:8]<br>MB[4]: ENET_MAC_ADDR2H[7:0]<br>MB[3]: ENET_MAC_ADDR2L[31:24]<br>MB[2]: ENET_MAC_ADDR2L[23:16]<br>MB[1]: ENET_MAC_ADDR2L[15:8]<br>MB[0]: ENET_MAC_ADDR2L[7:0] |
| 23:16 | Reserved | Must be kept at reset value. |

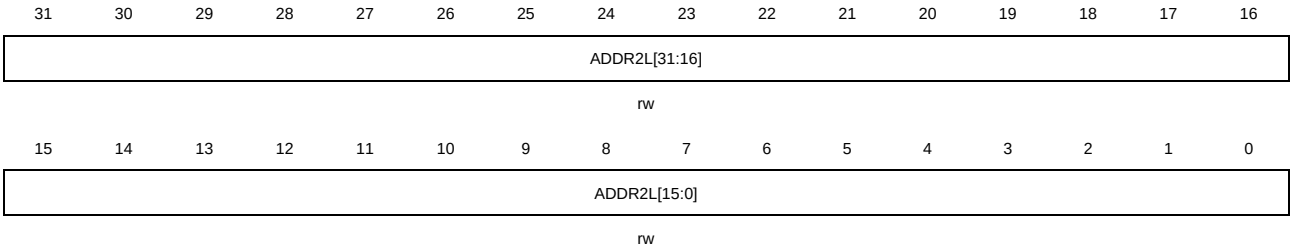
| | | |
|---|---|---|
| 15:0 | ADDR2H[15:0] | MAC address2 high 16-bit<br>This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address2. |

### 43.4.19. MAC address 2 low register (ENET_MAC_ADDR2L)

Address offset: 0x0054

Reset value: 0xFFFF FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



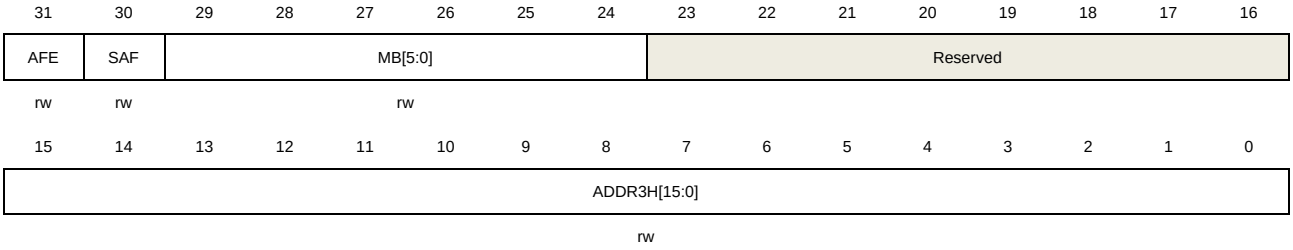

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR2L[31:0] | MAC address2 low 32-bit<br>This field contains the low 32-bit of the 6-byte MAC address2. |

### 43.4.20. MAC address 3 high register (ENET_MAC_ADDR3H)

Address offset: 0x0058

Reset value: 0x0000 FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | AFE | Address filter enable bit<br>0: MAC address3 is ignored by address filter for filtering<br>1: MAC address3 is used by address filter for perfect filtering |
| 30 | SAF | Source address filter bit<br>0: Comparing MAC address3 with the destination address field of the received frame<br>1: Comparing MAC address3 with the source address field of the received frame |
| 29:24 | MB[5:0] | Mask byte bits<br>If these bits is set, the destination address / source address corresponding byte of |

| | | |
|---|---|---|
| | | the received frame is not compared with MAC address3. Each bit controls one byte mask as follows:<br>MB[5]: ENET_MAC_ADDR3H[15:8]<br>MB[4]: ENET_MAC_ADDR3H[7:0]<br>MB[3]: ENET_MAC_ADDR3L[31:24]<br>MB[2]: ENET_MAC_ADDR3L[23:16]<br>MB[1]: ENET_MAC_ADDR3L[15:8]<br>MB[0]: ENET_MAC_ADDR3L[7:0] |
| 23:16 | Reserved | Must be kept at reset value. |
| 15:0 | ADDR3H[15:0] | MAC address3 high 16-bit<br>This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address3. |

## 43.4.21. MAC address 3 low register (ENET_MAC_ADDR3L)

Address offset: 0x005C

Reset value: 0xFFFF FFFF

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR3L[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR3L[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ADDR3L[31:0] | MAC address3 low 32-bit<br>This field contains the low 32-bit of the 6-byte MAC address3. |

## 43.4.22. MAC flow control threshold register (ENET_MAC_FCTH)

Address offset: 0x1080

Reset value: 0x0000 0015

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | RFD[2:0] | | | Reserved | RFA[2:0] | | |
| | | | | | | | | | rw | | | | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|

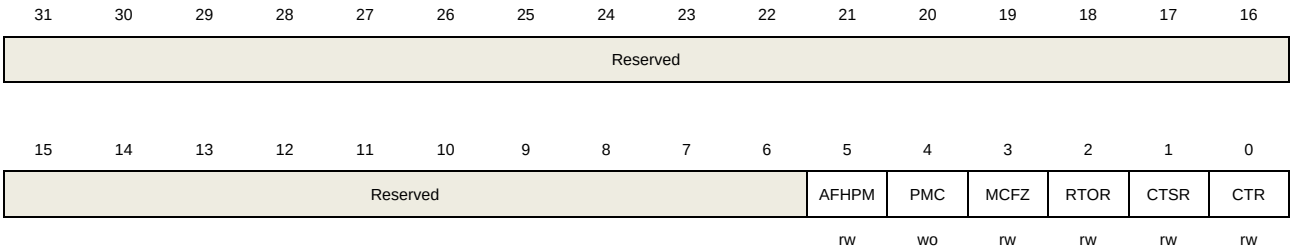
| | | |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6:4 | RFD[2:0] | Threshold of deactive flow control<br>This field configures the threshold of the deactive flow control. The value should always be less than the Threshold of active flow control value configured in bits[2:0]. When the value of the unprocessed data in RxFIFO is less than this value configured, the flow control function will deactive.<br>0x0: 256 bytes<br>0x1: 512 bytes<br>0x2: 768 bytes<br>0x3: 1024 bytes<br>0x4: 1280 bytes<br>0x5: 1536 bytes<br>0x6,0x7: 1792 bytes |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | RFA[2:0] | Threshold of active flow control<br>This field configures the threshold of the active flow control. If flow control function is enabled, when the value of the unprocessed data in RxFIFO is more than this value configured, the flow control function will active.<br>0x0: 256 bytes<br>0x1: 512 bytes<br>0x2: 768 bytes<br>0x3: 1024 bytes<br>0x4: 1280 bytes<br>0x5: 1536 bytes<br>0x6,0x7: 1792 bytes |

### 43.4.23. MSC control register (ENET_MSC_CTL)

Address offset: 0x0100

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5 | AFHPM | Almost full or half preset mode |

0: Preset all MSC counters to almost-half (0x7FFF FFF0) value
1: Preset all MSC counters to almost-full (0xFFFF FFF0) value
**Note:** This bit is valid only when PMC is set.

4 PMC Preset MSC counter
0: No effect
1: Preset MSC counters to a preset value. Preset value depends on AFHPM.

3 MCFZ MSC counter freeze bit
0: MSC counters are not frozen
1: Freezes all the MSC counters to their current value. RTOR bit can work on this frozen state.

2 RTOR Reset on read bit
0: The MSC counters are not reset after reading MSC counter
1: The MSC counters are reset to zero after read them

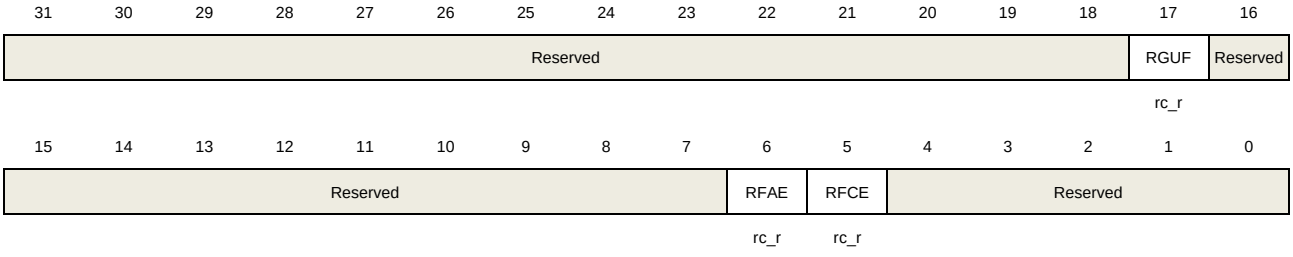
1 CTSR Counter stop rollover bit
0: The counters roll over to zero after they reached the maximum value
1: The counters do not roll over to zero after they reached the maximum value

0 CTR Counter reset bit
Cleared by hardware 1 clock after set.
This bit is cleared automatically after 1 clock cycle.
0: No effect
1: Reset all counters

## 43.4.24. MSC receive interrupt flag register (ENET_MSC_RINTF)

Address offset: 0x0104
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | RGUF | Received good unicast frames bit<br>0: Good unicast frame received counter is less than half of the maximum value<br>1: Good unicast frame received counter reaches half of the maximum value |

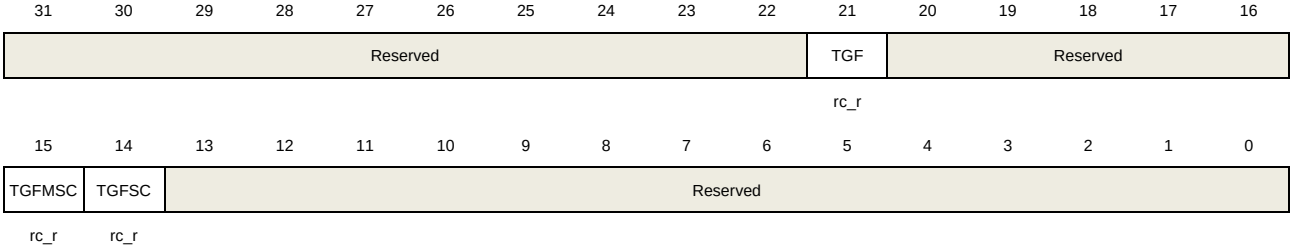
| | | |
|---|---|---|
| 16:7 | Reserved | Must be kept at reset value. |
| 6 | RFAE | Received frames alignment error bit<br>0: Alignment error frame received counter is less than half of the maximum value<br>1: Alignment error frame received counter reaches half of the maximum value |
| 5 | RFCE | Received frames CRC error bit<br>0: CRC error frame received counter is less than half of the maximum value<br>1: CRC error frame received counter reaches half of the maximum value |
| 4:0 | Reserved | Must be kept at reset value. |

### 43.4.25. MSC transmit interrupt flag register (ENET_MSC_TINTF)

Address offset: 0x0108

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



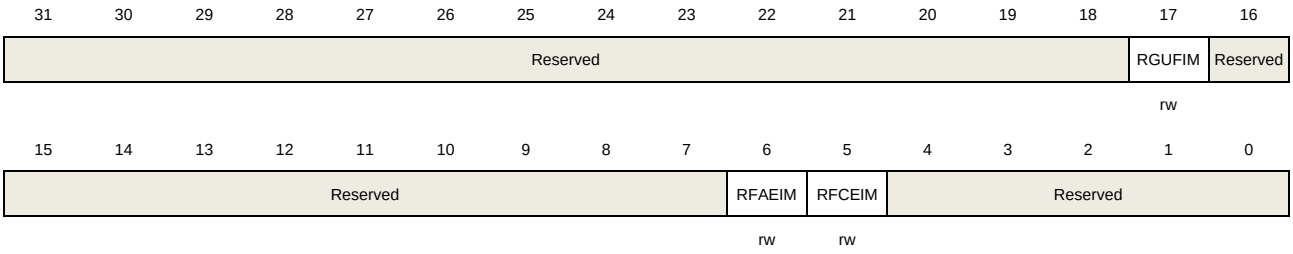

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21 | TGF | Transmitted good frames bit<br>0: Good frame transmitted counter is less than half of the maximum value<br>1: Good frame transmitted counter reaches half of the maximum value |
| 20:16 | Reserved | Must be kept at reset value. |
| 15 | TGFMSC | Transmitted good frames more single collision bit<br>0: Good frame after more than a single collision transmitted counter is less than half of the maximum value<br>1:Good frame after more than a single collision transmitted counter reaches half of the maximum value |
| 14 | TGFSC | Transmitted good frames single collision bit<br>0: Good frame after a single collision transmitted counter is less than half of the maximum value<br>1: Good frame after a single collision transmitted counter reaches half of the maximum value |
| 13:0 | Reserved | Must be kept at reset value. |

### 43.4.26. MSC receive interrupt mask register (ENET_MSC_RINTMSK)

Address offset: 0x010C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

The Ethernet MSC receive interrupt mask register maintains the masks for interrupts generated when receive statistic counters reach half their maximum value



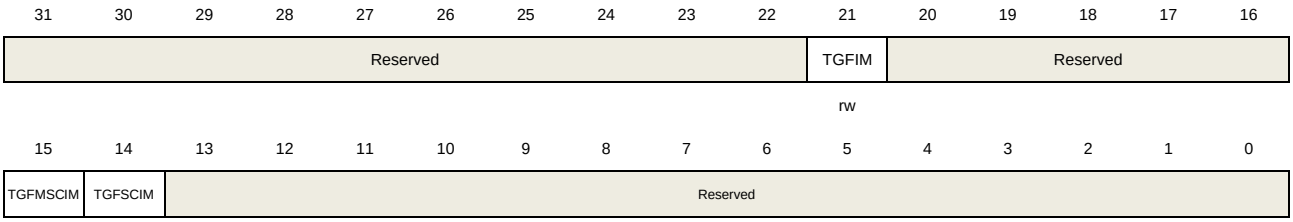

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | RGUFIM | Received good unicast frames interrupt mask bit<br>0: Unmask the interrupt when the RGUF bit is set<br>1: Mask the interrupt when RGUF bit is set |
| 16:7 | Reserved | Must be kept at reset value. |
| 6 | RFAEIM | Received frames alignment error interrupt mask bit<br>0: Unmask the interrupt when the RFAE bit is set<br>1: Mask the interrupt when the RFAE bit is set |
| 5 | RFCEIM | Received frame CRC error interrupt mask bit<br>0: Unmask the interrupt when RFCE bit is set<br>1: Mask the interrupt when the RFCE bit is set |
| 4:0 | Reserved | Must be kept at reset value. |

### 43.4.27. MSC transmit interrupt mask register (ENET_MSC_TINTMSK)

Address offset: 0x0110
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

The MSC transmit interrupt mask register configures the mask bits for interrupts generation



rw rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21 | TGFIM | Transmitted good frames interrupt mask bit<br>0: Unmask the interrupt when the TGF bit is set<br>1:Mask the interrupt when the TGF bit is set |
| 20:16 | Reserved | Must be kept at reset value. |
| 15 | TGFMSCIM | Transmitted good frames more single collision interrupt mask bit<br>0: Unmask the interrupt when the TGFMSC bit is set<br>1: Mask the interrupt when the TGFMSC bit is set |
| 14 | TGFSCIM | Transmitted good frames single collision interrupt mask bit<br>0: Unmask the interrupt when the TFGSC bit is set<br>1: Mask the interrupt when the TFGSC bit is set |
| 13:0 | Reserved | Must be kept at reset value. |

### 43.4.28. MSC transmitted good frames after a single collision counter register (ENET_MSC_SCCNT)

Address offset: 0x014C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register counts the number of successfully transmitted frames after a single collision in Half-duplex mode.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCC[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SCC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SCC[31:0] | Transmitted good frames single collision counter bits<br>These bits count the number of a transmitted good frames after only a single collision. |

### 43.4.29. MSC transmitted good frames after more than a single collision counter register (ENET_MSC_MSCCNT)

Address offset: 0x0150
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register counts the number of successfully transmitted frames after more than one single collision in Half-duplex mode.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSCC[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSCC[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | MSCC[31:0] | Transmitted good frames more one single collision counter bits<br>These bits count the number of a transmitted good frames after more than one single collision. |

### 43.4.30. MSC transmitted good frames counter register (ENET_MSC_TGFCNT)

Address offset: 0x0168
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register counts the number of good frames transmitted.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TGF[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TGF[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TGF[31:0] | Transmitted good frames counter bits<br>These bits count the number of transmitted good frames. |

### 43.4.31. MSC received frames with CRC error counter register (ENET_MSC_RFCECNT)

Address offset: 0x0194
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register counts the number of frames received with CRC error.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFCER[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFCER[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RFCER[31:0] | Received frames with CRC error counter bits<br>These bits count the number of receive frames with CRC error. |

### 43.4.32. MSC received frames with alignment error counter register (ENET_MSC_RFAECNT)

Address offset: 0x0198
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register counts the number of received frames with alignment error.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFAER[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RFAER[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RFAER[31:0] | Received frames alignment error counter bits<br>These bits count the number of receive frames with alignment error. |

### 43.4.33. MSC received good unicast frames counter register (ENET_MSC_RGUFCNT)

Address offset: 0x01C4
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register counts the number of good unicast frames received.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RGUF[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RGUF[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RGUF[31:0] | Received good unicast frames counter bits<br>These bits count the number of good unicast frames received. |

### 43.4.34. PTP time stamp control register (ENET_PTP_TSCTL)

Address offset: 0x0700
Reset value: 0x0000 2000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the generation and updating for timestamp.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | MAFEN | CKNT[1:0] | |
| | | | | | | | | | | | | | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MNMSEN | ETMSEN | IP4SEN | IP6SEN | ESEN | PFSV | SCROM | ARFSEN | Reserved | | TMSARU | TMSITEN | TMSSTU | TMSSTI | TMSFCU | TMSEN |
| rw | rw | rw | rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18 | MAFEN | MAC address filter enable for PTP frame<br>0: No effect<br>1: Enable MAC address1-3 to filter the PTP frame when received frame's type field is 0x88f7 |
| 17:16 | CKNT[1:0] | Clock node type for time stamp<br>0x0: Type of ordinary clock |

| | | |
|---|---|---|
| | | 0x1: Type of boundary clock<br>0x2: Type of end-to-end transparent clock<br>0x3: Type of peer-to-peer transparent clock |
| 15 | MNMSEN | Received master node message snapshot enable<br>This bit is valid only when CKNT=0x0 or 0x1.<br>0: Snapshot is only taken for slave node message<br>1: Snapshot is only take for master node message |
| 14 | ETMSEN | Received event type message snapshot enable<br>0: All type messages are taken snapshot except Announce, Management and Signaling message<br>1: Only event type messages (SYNC, DELAY_REQ, PDELAY_REQ and PDELAY_RESP) are taken snapshot |
| 13 | IP4SEN | Received IPv4 snapshot enable<br>0: Do not take snapshot for IPv4 frame<br>1: Take snapshot for IPv4 frame |
| 12 | IP6SEN | Received IPv6 snapshot enable<br>0: Do not take snapshot for IPv6 frame<br>1: Take snapshot for IPv6 frame |
| 11 | ESEN | Received Ethernet snapshot enable<br>0: Do not take snapshot when received non type frame<br>1: Take snapshot when received non type frame |
| 10 | PFSV | PTP frame snooping version<br>0: Version 1 (Revision of IEEE STD. 1588-2002 / 1588-2008)<br>1: Version 2 (Revision of IEEE STD. 1588-2008) |
| 9 | SCROM | Subsecond counter rollover mode<br>0: Binary rollover mode. Subsecond rollovers when reach 0x7FFF_FFFF<br>1: Digital rollover mode. Subsecond rollovers when reach 0x3B9A_C9FF (0d999_999_999) |
| 8 | ARFSEN | All received frames snapshot enable<br>0: Not all received frames are taken snapshot<br>1: All received frames are taken snapshot |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | TMSARU | Time stamp addend register update bit<br>0: The value of ENET_PTP_TSADDEND register is not updated to the PTP block for fine correction<br>1: The value of ENET_PTP_TSADDEND register is updated to the PTP block for fine correction<br>**Note**: Before user set it, the TMSARU bit must be read as 0. When update is finish, the TMSARU bit is cleared. |

| | | |
|---|---|---|
| 4 | TMSITEN | Timestamp interrupt trigger enable bit<br>0: Disable timestamp interrupt<br>1: When the system time is no less than the value in ENET_PTP_ETH and ENET_PTP_ETL registers, a timestamp interrupt is generated.<br>**Note**: After the timestamp trigger interrupt happened the TMSITEN bit is cleared. |
| 3 | TMSSTU | Timestamp system time update bit<br>Both the TMSSTU and TMSSTI bits must be read as zero before application set this bit.<br>0: Not update the system time<br>1: Update the system time with the value in the ENET_PTP_TSUH and ENET_PTP_TSUL registers. It is cleared by hardware when the update finished. |
| 2 | TMSSTI | Timestamp system time initialize bit<br>This bit must be read as 0 before application set it.<br>0: The system time is maintained without any change<br>1: Initializing the system time with the value in ENET_PTP_TSUH and ENET_PTP_TSUL registers. It is cleared by hardware when the initialization finished. |
| 1 | TMSFCU | Timestamp fine or coarse update bit<br>0: The system timestamp uses the coarse method for updating<br>1:The system timestamp uses the fine method for updating |
| 0 | TMSEN | Timestamp enable bit<br>0: Disable timestamp function<br>1: Enable timestamp function for transmit and receive frames<br>**Note:** After setting this to 1, application must initialize the system time. |

**Table 43-8. Supported time stamp snapshot with PTP register configuration**

| **CKNT (Bit 17:16)** | **0X** | | | **10** | | **11** | |
|---|---|---|---|---|---|---|---|
| **MNMSEN (Bit 15)** | X(*) | 1 | 0 | X | | | |
| **ETMSEN (Bit 14)** | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| **Supported message type for snapshot** | SYNC FOLLOW_UP DELAY_REQ DELAY_RESP | DELAY_REQ | SYNC | SYNC FOLLOW_UP DELAY_REQ DELAY_RESP | SYNC FOLLOW_UP | SYNC FOLLOW_UP DELAY_REQ DELAY_RESP PDELAY_REQ PDELAY_RESP | SYNC PDELAY_REQ PDELAY_RESP |

*: means do not care

### 43.4.35. PTP subsecond increment register (ENET_PTP_SSINC)

Address offset: 0x0704
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the 8-bit value for the incrementing subsecond register. In coarse mode, this value is added to the system time every HCLK clock cycle. In fine mode, this value is added to the system time when the accumulator reaches overflow.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | STMSSI[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | STMSSI[7:0] | System time subsecond increment bits<br>In every update operation, these bits are added to the subsecond value of system time. |

### 43.4.36. PTP time stamp high register (ENET_PTP_TSH)

Address offset: 0x0708
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STMS[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

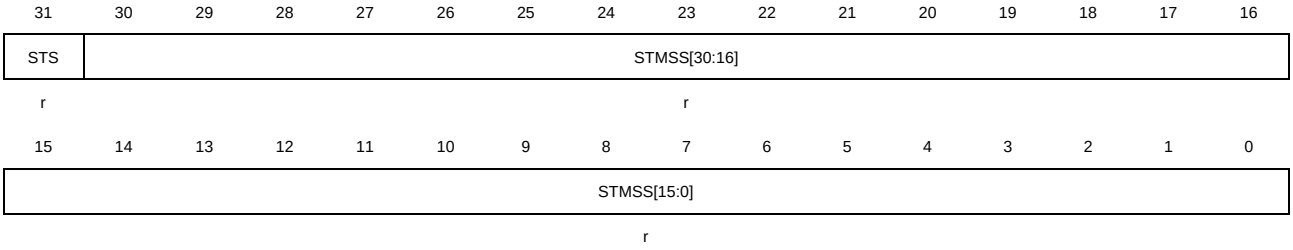
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STMS[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | STMS[31:0] | System time second bits<br>These bits show the current second of the system time. |

### 43.4.37. PTP time stamp low register (ENET_PTP_TSL)

Address offset: 0x070C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



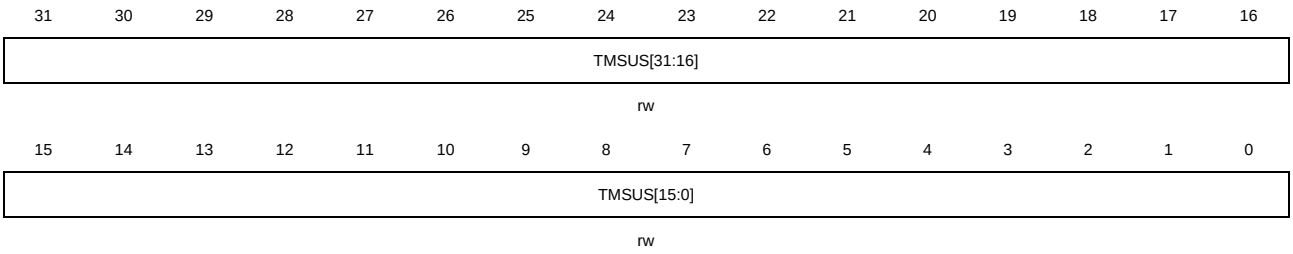

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | STS | System time sign bit<br>0: Time value is positive<br>1: Time value is negative |
| 30:0 | STMSS[30:0] | System time subseconds bits<br>These bits show the current subsecond of the system time with 0.46 ns accuracy |

## 43.4.38. PTP time stamp update high register (ENET_PTP_TSUH)

Address offset: 0x0710
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the high 32-bit of the time to be written to, added to, or subtracted from the system time value. The timestamp update registers (high and low) initialize or update the system time maintained by the MAC core. Application must write both of these registers before setting the TMSSTI or TMSSTU bits in the timestamp control register.

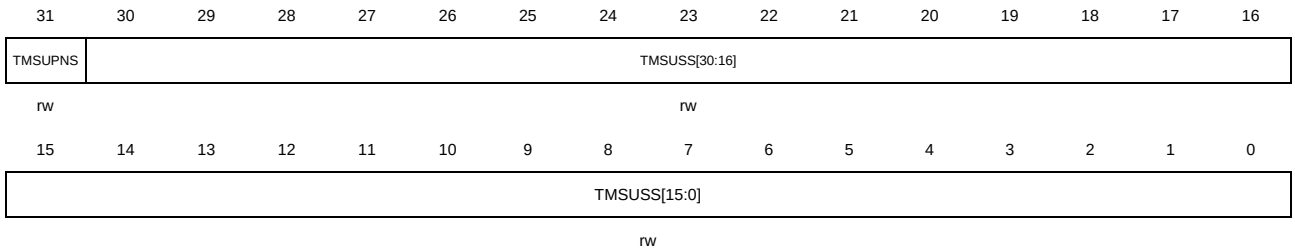



| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TMSUS[31:0] | Time stamp update second bits<br>These bits are used for initializing or adding / subtracting to second of the system time. |

## 43.4.39. PTP time stamp update low register (ENET_PTP_TSUL)

Address offset: 0x0714
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



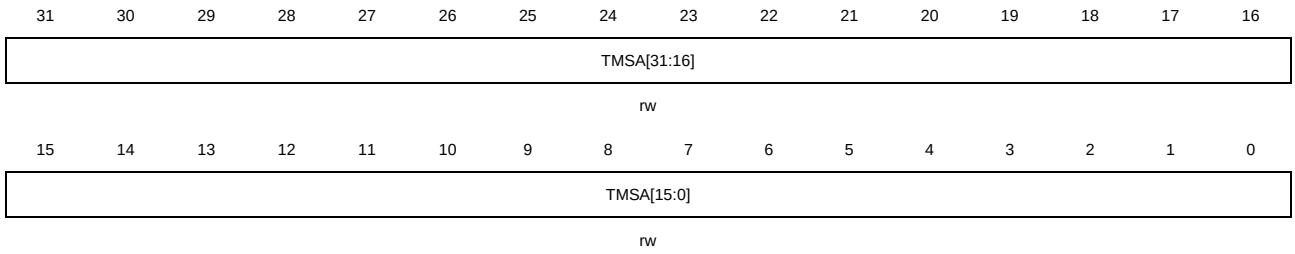

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TMSUPNS | Timestamp update positive or negative sign bit<br>When TMSSTI is set, this bit must be 0.<br>0: Timestamp update value is added to system time<br>1: Timestamp update value is subtracted from system time |
| 30:0 | TMSUSS[30:0] | Timestamp update subsecond bits<br>These bits are used for initializing or adding / subtracting to subsecond of the system time. |

## 43.4.40. PTP time stamp addend register (ENET_PTP_TSADDEND)

Address offset: 0x0718
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register value is used only in fine update mode for adjusting the clock frequency. This register value is added to a 32-bit accumulator in every clock cycle and the system time updates when the accumulator reaches overflow.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TMSA[31:0] | Time stamp addend bits<br>In order to achieve time synchronization, the value of TMSA[31:0] is added to the accumulator register. |

## 43.4.41. PTP expected time high register (ENET_PTP_ETH)

Address offset: 0x071C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETSH[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETSH[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ETSH[31:0] | Expected time high bits<br>These bits store the expected target second time. |

## 43.4.42. PTP expected time low register (ENET_PTP_ETL)

Address offset: 0x0720

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETSL[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ETSL[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ETSL[31:0] | Expected time low bits<br>These bits store the expected target nanosecond time (signed). |

## 43.4.43. PTP time stamp flag register (ENET_PTP_TSF)

Address offset: 0x0728

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | TTM | TSSCO |
| | | | | | | | | | | | | | | ro | ro |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |

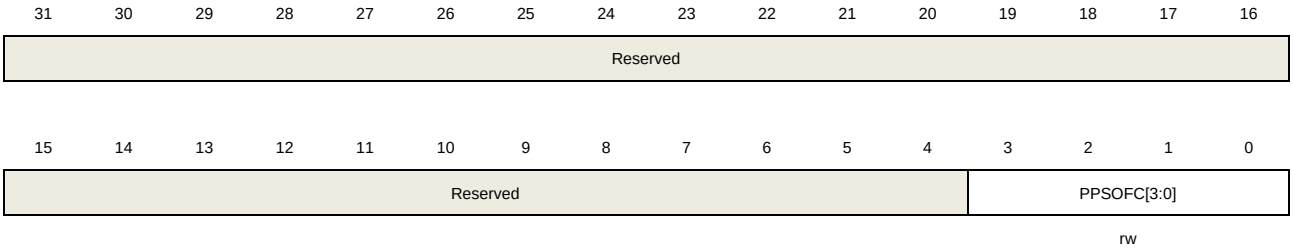
| | | |
|---|---|---|
| 1 | TTM | Target time match bit<br>0: System time is not equal or greater than expected time.<br>1: System time is equal or greater than expected time<br>**Note:** Reading ENET_PTP_TSF register will clear this bit. |
| 0 | TSSCO | Timestamp second counter overflow bit<br>0: Timestamp second counter has not overflowed<br>1: Timestamp second counter is greater than 0xFFFF FFFF |

## 43.4.44. PTP PPS control register (ENET_PTP_PPSCTL)

Address offset: 0x072C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



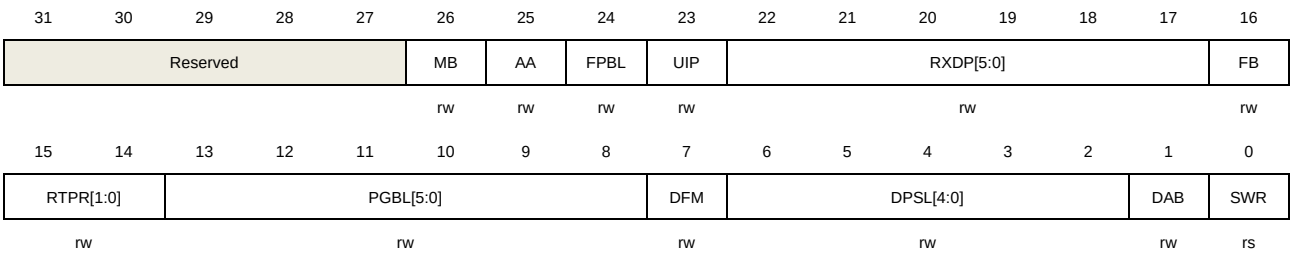

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value. |
| 3:0 | PPSOFC | PPS output frequency configure<br>0x0: 1Hz (Pulse width: 125ms for binary rollover, 100ms for digital rollover)<br>0x1: 2Hz (Pulse width: 50% duty cycle for binary rollover)<br>0x2: 4Hz (Pulse width: 50% duty cycle for binary rollover)<br>….<br>0xF: 32768 ($2^{15}$) Hz (Pulse width: 50% duty cycle for binary rollover)<br>**Note:** If digital rollover is selected, only PPSOFC=0 is recommended. |

## 43.4.45. DMA bus control register (ENET_DMA_BCTL)

Address offset: 0x1000

Reset value: 0x0002 0101

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |
| 26 | MB | Mixed burst<br>0: AHB master interface only transfer fixed burst length with 16 and below<br>1: AHB master interface will transfer burst length greater than 16 with INCR<br>**Note:** MB and FB should be and must be only one of bit is set. |
| 25 | AA | Address-aligned bit<br>0: Disable address-aligned<br>1: Enabled address-aligned. If the FB=1, all AHB interface address is aligned to the start address LS bits (bit 1 to 0). If the FB=0, the AHB interface first access address (accessing the data buffer's start address) is not aligned, but subsequent burst access addresses are aligned to the address. |
| 24 | FPBL | Four times PGBL mode bit<br>0: The PGBL value programmed (bits[22:17] and bits[13:8]) for the DMA data number of beats to be transferred<br>1: Multiple the PGBL value programmed (bits[22:17] and bits[13:8]) four times for the DMA data number of beats to be transferred |
| 23 | UIP | Use independent PGBL bit<br>0: The PGBL value in bits[13:8] is applicable for both TxDMA and RxDMA engines<br>1: The RxDMA uses the RXDP[5:0] bits as burst length while the PGBL[5:0] is used by TxDMA |
| 22:17 | RXDP[5:0] | RxDMA PGBL bits<br>If UIP=0, these bits are not valid. Only when UIP=1, these bits is configured for the maximum number of beats to be transferred in one RxDMA transaction.<br>0x01: max beat number is 1<br>0x02: max beat number is 2<br>0x04: max beat number is 4<br>0x08: max beat number is 8<br>0x10: max beat number is 16<br>0x20: max beat number is 32<br>Other: Reserved |
| 16 | FB | Fixed burst bit<br>0: Both SINGLE and INCR burst transfer operations can be used by AHB<br>1: Only SINGLE, INCR4, INCR8 or INCR16 can be used by AHB, while in the start of normal burst transfer.<br>**Note:** MB and FB should be and must be only one of bit is set. |
| 15:14 | RTPR[1:0] | RxDMA and TxDMA transfer priority ratio bits<br>These bits indicate the access ratio between RxDMA and TxDMA.<br>0x0: RxDMA : TxDMA = 1:1<br>0x1: RxDMA : TxDMA = 2:1 |

| | | |
|---|---|---|
| | | 0x2: RxDMA : TxDMA = 3:1<br>0x3: RxDMA : TxDMA = 4:1<br>**Note:** This bit is valid only when the arbitration mode is Round-robin (DAB=0). |
| 13:8 | PGBL[5:0] | Programmable burst length bits<br>These bits indicate the maximum number of beats to be transferred in one DMA transaction. When UIP=1, the PGBL value is only used for TxDMA. When UIP=0, the PGBL value is used for both TxDMA and RxDMA.<br>0x01: max beat number is 1<br>0x02: max beat number is 2<br>0x04: max beat number is 4<br>0x08: max beat number is 8<br>0x10: max beat number is 16<br>0x20: max beat number is 32<br>Other: Reserved. |
| 7 | DFM | Descriptor format mode<br>0: Normal mode descriptor<br>1: Enhanced mode descriptor |
| 6:2 | DPSL[4:0] | Descriptor skip length bit<br>These bits are valid only between two ring mode descriptors. They define the number of words (32-bit) to skip between two ring descriptors. DPSL[4:0] represents the address difference from the end of the current descriptor to the beginning of the next descriptor. If the value of DPSL[4:0] is 0, the DMA taking the descriptor table as contiguous. |
| 1 | DAB | DMA arbitration bit<br>This bit indicates the arbitration mode between RxDMA and TxDMA.<br>0: Round-robin mode and DMA access priority is given in RTPR<br>1: Fixed mode. RxDMA has higher priority than TxDMA |
| 0 | SWR | Software reset bit<br>This bit can reset all core internal registers located in CLK_TX and CLK_RX.<br>It is cleared by hardware when the reset operation is complete in all clock domains.<br>**Note:** Application must make sure this bit is 0 before writing any MAC core registers.<br>0: Core and inner register are not in reset state<br>1: Reset all core internal registers |

### 43.4.46. DMA transmit poll enable register (ENET_DMA_TPEN)

Address offset: 0x1004
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register is used by the application to make the TxDMA controller poll the transmit descriptor table. The TxDMA controller can go into suspend state because of an underflow

error in a transmitted frame or the descriptor unavailable (DAV=0). Application can write any value into this register for attempting to re-fetch the current descriptor.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TPE[31:16] | | | | | | | | | | | | | | | |
| rw_wt | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TPE[15:0] | | | | | | | | | | | | | | | |
| rw_wt | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TPE[31:0] | Transmit poll enable bits<br>Writing to this register with any value makes DMA read the current descriptor address which is indicated in ENET_DMA_CTDADDR register. If the fetched current descriptor is available (DAV=1), DMA exits suspend state and resumes working. If the fetched current descriptor is unavailable (DAV=0), the DMA returns to suspend state again and the TBU bit in ENET_DMA_STAT register will be set. |

## 43.4.47. DMA receive poll enable register (ENET_DMA_RPEN)

Address offset: 0x1008
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register is used by the application to make the RxDMA controller poll the receive descriptor table. Writing to this register makes the RxDMA controller exit suspend state.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RPE[31:16] | | | | | | | | | | | | | | | |
| rw_wt | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RPE[15:0] | | | | | | | | | | | | | | | |
| rw_wt | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RPE[31:0] | Receive poll enable bits<br>Writing to this register with any value makes DMA read the current descriptor address which is indicated in ENET_DMA_CRDADDR register. If the fetched current descriptor is available (DAV=1), DMA exits suspend state and resumes working. If the fetched current descriptor is unavailable (DAV=0), the DMA returns to suspend state again and the RBU bit in ENET_DMA_STAT register will be set. |

## 43.4.48. DMA receive descriptor table address register (ENET_DMA_RDTADDR)

Address offset: 0x100C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register points to the start of the receive descriptor table. The descriptor table is located in the physical memory space and must be word-aligned. This register can only be written when RxDMA controller is in stop state. Before starting RxDMA reception process, this register must be configured correctly.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRT[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SRT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SRT[31:0] | Start address of receive table bits<br>These bits indicate the start address of the receive descriptor table. SRT[1:0] are internally taken as zero so SRT[1:0] are read only. |

## 43.4.49. DMA transmit descriptor table address register (ENET_DMA_TDTADDR)

Address offset: 0x1010
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register points to the start of the transmit descriptor table. The descriptor table is located in the physical memory space and must be word-aligned. This register can only be written when TxDMA controller is in stop state. Before starting TxDMA transmission process, this register must be configured correctly.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STT[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| STT[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | STT[31:0] | Start address of transmit table bits<br>These bits indicate the start address of the transmit descriptor table. STT[1:0] are internally taken as zero so STT[1:0] are read only. |

### 43.4.50. DMA status register (ENET_DMA_STAT)

Address offset: 0x1014

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register contains all the status bits that the DMA controller recorded. Writing 1 to meaningful bits in this register clears them but writing 0 has no effect. Each bit (bits[16:0]) can be masked by masking the corresponding bit in the ENET_DMA_INTEN register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | TST | WUM | MSC | Reserved | EB[2:0] | | | TP[2:0] | | | RP[2:0] | | | NI |
| | | r | r | r | | r | | | r | | | r | | | rc_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AI | ER | FBE | Reserved | | ET | RWT | RPS | RBU | RS | TU | RO | TJT | TBU | TPS | TS |
| rc_w1 | rc_w1 | rc_w1 | | | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value. |
| 29 | TST | Timestamp trigger status bit<br>This bit indicates a timestamp event occurred. It is cleared by application through clearing TMST bit. If the corresponding interrupt mask bit is reset, an interrupt is generated.<br>0: Timestamp event has not occurred<br>1: Timestamp event has occurred |
| 28 | WUM | WUM status bit<br>This bit indicates a WUM event occurred. It is cleared when both two source event status bits are cleared. If the corresponding interrupt mask bit is reset, an interrupt is generated.<br>0: WUM event has not occurred<br>1: WUM event has occurred |
| 27 | MSC | MSC status bit<br>This bit indicates a MSC event occurred. It is cleared when all of event sources are cleared. If the corresponding interrupt mask bit is reset, an interrupt is generated.<br>0: MSC event has not occurred<br>1: MSC event has occurred |
| 26 | Reserved | Must be kept at reset value. |
| 25:23 | EB[2:0] | Error bits status bit<br>When FBE=1, these bits decode the type of error that caused a bus response error on AHB bus.<br>EB[0]:<br>1: Error occurs while TxDMA transfer data |

| | | |
|---|---|---|
| | | 0: Error occurs while RxDMA transfer data<br>EB[1]:<br>1: Error occurs while read transfer<br>0: Error occurs while write transfer<br>EB[2]:<br>1: Error occurs while access descriptor<br>0: Error occurs while access data buffer |
| 22:20 | TP[2:0] | Transmit process state bit<br>These bits decode the TxDMA state.<br>0x0: Stopped; Issuing transmit command which is Reset or Stop.<br>0x1: Running; Fetching the transfer descriptor that belongs to transmit.<br>0x2: Running; Waiting for status<br>0x3: Running; Queuing it to TxFIFO after reading transmit packet data from host memory buffer.<br>0x4, 0x5: Reserved<br>0x6: Suspended; Unavailable of transmit descriptor or underflow of transmit buffer.<br>0x7: Running; Closing the descriptor that belongs to transmit. |
| 19:17 | RP[2:0] | Receive process state bit<br>These bits decode the RxDMA state.<br>0x0: Stopped; Issuing receive command which is Reset or Stop.<br>0x1: Running; Fetching the transfer descriptor that belongs to receive.<br>0x2: Reserved<br>0x3: Running; Waiting for the packet that belongs to receive.<br>0x4: Suspended: Unavailable of receive descriptor<br>0x5: Running: Closing the descriptor that belongs to receive.<br>0x6: Reserved<br>0x7: Running; Transferring it to host memory after reading the receive packet data from RxFIFO. |
| 16 | NI | Normal interrupt summary<br>The NI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register:<br>TS: Interrupt of transmit<br>TBU: Unavailable of transmit buffer<br>RS: Interrupt of receive<br>ER: Interrupt of early receive<br>**Note:** Each time when this bit is set, application must cleared its source bit by writing 1 to that bit. |
| 15 | AI | Abnormal interrupt summary bit<br>The AI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register:<br>TPS: Halt of transmit process |

| | | |
|---|---|---|
| | | TJT: Timeout of transmit jabber<br>RO: Overflow of receive FIFO<br>TU: Underflow transmit<br>RBU: Receive buffer<br>RPS: Unavailable of receive process stopped<br>RWT: Timeout of receive watchdog<br>ET: Interrupt of early transmit<br>FBE: Error of fatal bus error<br>**Note:** Each time when this bit is set, application must cleared its source bit by writing 1 to that bit. |
| 14 | ER | Early receive status bit<br>This bit is automatically cleared when the RS bit is set.<br>0: The first buffer has not been filled<br>1: The first buffer has filled with received frame |
| 13 | FBE | Fatal bus error status bit<br>This bit indicates a response error on AHB interface is occurred and the error type can be decoded by EB[2:0] bits.<br>0: Bus error has not occurred<br>1: A bus error occurred and the corresponding DMA stops all operations |
| 12:11 | Reserved | Must be kept at reset value. |
| 10 | ET | Early transmit status bit<br>0: The frame to be transmitted has not fully transferred into the TxFIFO<br>1: The frame to be transmitted has fully transferred into the TxFIFO |
| 9 | RWT | Receive watchdog timeout status bit<br>0: No received a frame with a length greater than 2048 bytes<br>1: A frame with a length greater than 2048 bytes is received |
| 8 | RPS | Receive process stopped status bit<br>0: The receive process is not in stop state<br>1: The receive process is in stop state |
| 7 | RBU | Receive buffer unavailable status bit<br>0: The DAV bit in fetched next receive descriptor is set<br>1: The DAV bit in fetched next receive descriptor is reset and RxDMA enters suspend state |
| 6 | RS | Receive status bit<br>0: Frame reception has not completed<br>1: Frame reception has completed |
| 5 | TU | Transmit underflow status bit<br>0: Underflow error has not occurred during frame transmission<br>1: The TxFIFO encountered an underflow error during frame transmission and |

| | | |
|---|---|---|
| | | entered suspend state |
| 4 | RO | Receive overflow status bit<br>0: Receive overflow error has not occurred during frame reception<br>1: The RxFIFO encountered an overflow error during frame reception. If a part of frame data has transferred to the memory, the overflow status in OERR bit is also set. |
| 3 | TJT | Transmit jabber timeout status bit<br>0: Transmit jabber timeout has not occurred during frame transmission<br>1: The transmit jabber timer expired. The TxDMA controller cancels the current transmission and enters stop state. This also causes JT bit in Transmit Descriptor0 set. |
| 2 | TBU | Transmit buffer unavailable status bit<br>0: The DAV bit in fetched next transmit descriptor is set<br>1: The DAV bit in fetched next transmit descriptor is reset and TxDMA enters suspend state |
| 1 | TPS | Transmit process stopped status bit<br>0: The transmission is not in stop state<br>1: The transmission is in stop state |
| 0 | TS | Transmit status bit<br>This bit can only be set when both LSG and INTC are set in Transmit Descriptor0.<br>0: Current frame transmission is not finished<br>1: Current frame transmission is finished. |

## 43.4.51. DMA control register (ENET_DMA_CTL)

Address offset: 0x1018
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures both the transmitting and receiving operation modes and commands. This register should be written at last during the process of DMA initialization.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | DTCERFD | RSFD | DAFRF | Reserved | | TSFD | FTF | Reserved | | | TTHC[2] |
| | | | | | rw | rw | rw | | | rw | rs | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TTHC[1:0] | | STE | Reserved | | | | | FERF | FUF | Reserved | RTHC[1:0] | | OSF | SRE | Reserved |
| rw | | rw | | | | | | rw | rw | | rw | | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:27 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 26 | DTCERFD | Dropping of TCP / IP checksum error frames disable bit<br>0: All error frames will be dropped when FERF=0<br>1: The received frame with only payload error but no other errors will not be dropped. |
| 25 | RSFD | Receive Store-and-Forward bit<br>0: The RxFIFO operates in Cut-Through mode. The forwarding threshold depends on the RTHC bits<br>1: The RxFIFO operates in Store-and-Forward mode. The RTHC bits are don't care and the frame forwarding starts after the whole frame has pushed into RxFIFO. |
| 24 | DAFRF | Disable flushing of received frames bit<br>0: The RxDMA flushes all frames because of unavailable receive descriptor<br>1: The RxDMA does not flush any frames even though receive descriptor is unavailable |
| 23:22 | Reserved | Must be kept at reset value. |
| 21 | TSFD | Transmit Store-and-Forward bit<br>0: The TxFIFO operates in Cut-Through mode. The TTHC bits in ENET_DMA_CTL register defines the start popping time from TxFIFO.<br>1: The TxFIFO operates in Store-and-Forward mode. Transmission on interface starts after the full frame has been pushed into the TxFIFO. The TTHC bits are don't care in this mode.<br>**Note:** This bit can be changed when transmission is in stop state. |
| 20 | FTF | Flush transmit FIFO bit<br>This bit can be set by application to reset TxFIFO inner control register and logic. If set, all data in TxFIFO are flushed. It is cleared by hardware after the flushing operation is finish.<br>**Note:** Before this bit is reset, this register (ENET_DMA_CTL) must not be written. |
| 19:17 | Reserved | Must be kept at reset value. |
| 16:14 | TTHC[2:0] | Transmit threshold control bit<br>These bits control the start transmitting byte threshold of the TxFIFO.<br>When TSFD=1, these bits are ignored.<br>0x0: 64<br>0x1: 128<br>0x2: 192<br>0x3: 256<br>0x4: 40<br>0x5: 32<br>0x6: 24<br>0x7: 16 |
| 13 | STE | Start / stop transmission enable bit<br>0: The TxDMA controller will enter stop state after transmitting complete if the current frame is being transmitted. After complete transmitting, the next descriptor |

| | | |
|---|---|---|
| | | address will become current descriptor address for the address pointer. If the TxDMA controller is in suspend state, reset this bit make the controller entering stop state.<br>1: The TxDMA controller will enter running state. TxDMA controller fetches current descriptor address for frame transmitting. Transmit descriptor's fetching can either from base address in ENET_DMA_TDTADDR register or from the pointer position when transmission was stopped previously. If the DAV bit of current descriptor is reset, TxDMA controller enters suspend state and the TBU bit will be set. This bit should be set after all other DMA registers have been configured otherwise the action of TxDMA is unpredictable. |
| 12:8 | Reserved | Must be kept at reset value. |
| 7 | FERF | Forward error frames bit<br>0: When RxFIFO is in Cut-Through mode (RSFD=0), if frame error (CRC error, collision error, checksum error, watchdog timeout, overflow error) is detected before popping RxFIFO data to memory, RxFIFO drops this error frame. But if frame error is detected after popping RxFIFO data to memory, RxFIFO will not drop this frame data. When RxFIFO is in Store-and-Forward mode, once frame error is detected during reception the RxFIFO drops this frame.<br>1: All frame received with error except runt error are forwarded to memory |
| 6 | FUF | Forward undersized good frames bit<br>0: The RxFIFO drops all frames whose length is less than 64 bytes. However, if this frame has already started forwarding (may due to lower value of receive threshold in Cut-Through mode), the whole frame will be forwarded.<br>1: The RxFIFO forwards received frame whose frame length is less than 64 bytes but without any other error |
| 5 | Reserved | Must be kept at reset value. |
| 4:3 | RTHC[1:0] | Receive threshold control bit<br>These bits control the threshold bytes of the RxFIFO.<br>**Note:** These bits are valid only when the RSFD=0 and are ignored when the RSFD=1.<br>0x0: 64<br>0x1: 32<br>0x2: 96<br>0x3: 128 |
| 2 | OSF | Operate on second frame bit<br>0: The TxDMA controller process the second transmit frame after the status of the first frame is written back to descriptor<br>1: The TxDMA controller process the second transmit frame after pushed all first frame data into TxFIFO but before the status of the first frame is written back to descriptor |
| 1 | SRE | Start / stop receive enable bit |

0: The RxDMA controller will enter stop state after transfer complete if current received frame is transmitting to memory by RxDMA. After transfer complete, the next descriptor address in the receive table will become the current descriptor address when restart the RxDMA controller. Only RxDMA controller is in running state or suspend state, this bit can be reset by application.
1: The RxDMA controller will enter running state. RxDMA controller fetches receive descriptor from receive descriptor table for receiving frames. The descriptor address can either from current address in the ENET_DMA_RDTADDR register or the address after previous frame stopped by application. If the DAV bit in fetched descriptor is reset, RxDMA controller will enter suspend state and RBU bit will be set. Setting this bit can only when RxDMA controller is in stop state or suspend state. This bit should be set after all other DMA registers have been configured otherwise the action of RxDMA is unpredictable.

| | | |
|---|---|---|
| 0 | Reserved | Must be kept at reset value. |

## 43.4.52. DMA interrupt enable register (ENET_DMA_INTEN)

Address offset: 0x101C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register configures the interrupts which are reflected in ENET_DMA_STAT register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | NIE |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AIE | ERIE | FBEIE | Reserved | | ETIE | RWTIE | RPSIE | RBUIE | RIE | TUIE | ROIE | TJTIE | TBUIE | TPSIE | TIE |
| rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | NIE | Normal interrupt summary enable bit<br>0: Disable normal interrupt<br>1: Enable normal interrupt<br>This bit enables the following bits:<br>TS: Interrupt of transmit<br>TBU: Unavailable transmit buffer<br>RS: Interrupt of receive<br>ER: Interrupt of Early receive |
| 15 | AIE | Abnormal interrupt summary enable bit<br>0: Disable abnormal interrupt<br>1: Enable abnormal interrupt |

| | | |
|---|---|---|
| | | This bit enables the following bits:<br>TPS: Halt of transmit process<br>TJT: Timeout of transmit jabber<br>RO: Overflow of receive FIFO<br>TU: Underflow transmit<br>RBU: Receive buffer<br>RPS: Unavailable of receive process stopped<br>RWT: Timeout of receive watchdog<br>ET: Interrupt of early transmit<br>FBE: Error of fatal bus error |
| 14 | ERIE | Early receive interrupt enable bit<br>0: Disable early receive interrupt<br>1: Enable early receive interrupt |
| 13 | FBEIE | Fatal bus error interrupt enable bit<br>0: Disable fatal bus error interrupt<br>1: Enable fatal bus error interrupt |
| 12:11 | Reserved | Must be kept at reset value. |
| 10 | ETIE | Early transmit interrupt enable bit<br>0: Disable early transmit interrupt<br>1: Enable early transmit interrupt |
| 9 | RWTIE | Receive watchdog timeout interrupt enable bit<br>0: Disable receive watchdog timeout interrupt<br>1: Enable receive watchdog timeout interrupt |
| 8 | RPSIE | Receive process stopped interrupt enable bit<br>0: Disable receive stopped interrupt<br>1: Enable receive stopped interrupt |
| 7 | RBUIE | Receive buffer unavailable interrupt enable bit<br>0: Disable receive buffer unavailable interrupt<br>1: Enable receive buffer unavailable interrupt |
| 6 | RIE | Receive interrupt enable bit<br>0: Disable receive interrupt<br>1: Enable receive interrupt |
| 5 | TUIE | Transmit underflow interrupt enable bit<br>0: Disable underflow interrupt<br>1: Enable underflow interrupt |
| 4 | ROIE | Receive overflow interrupt enable bit<br>0: Disable overflow interrupt<br>1: Enable overflow interrupt |
| 3 | TJTIE | Transmit jabber timeout interrupt enable bit |

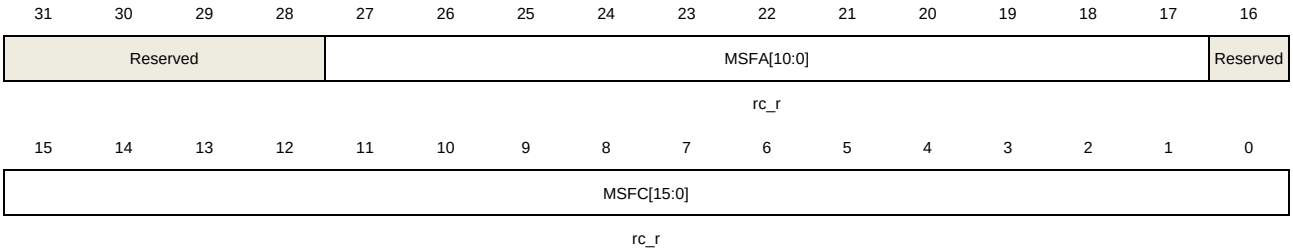
| | | |
|---|---|---|
| | | 0: Disable transmit jabber timeout interrupt<br>1: Enable transmit jabber timeout interrupt |
| 2 | TBUIE | Transmit buffer unavailable interrupt enable bit<br>0: Disable transmit buffer unavailable interrupt<br>1: Enable transmit buffer unavailable interrupt |
| 1 | TPSIE | Transmit process stopped interrupt enable bit<br>0: Disable transmission stopped interrupt<br>1: Enable transmission stopped interrupt |
| 0 | TIE | Transmit interrupt enable bit<br>0: Disable transmit interrupt<br>1: Enable transmit interrupt |

### 43.4.53. DMA missed frame and buffer overflow counter register (ENET_DMA_MFBOCNT)

Address offset: 0x1020
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

There are two counters designed in DMA controller for tracking the number of missed frames during receiving. The counter value can be read from this register for debug purpose.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:17 | MSFA[10:0] | Missed frames by the application bits<br>These bits indicate the number of frames dropped by RxFIFO. |
| 16 | Reserved | Must be kept at reset value. |
| 15:0 | MSFC[15:0] | Missed frames by the controller bits<br>These bits indicate the number of frames missed by the RxDMA controller because of the unavailable receive buffer. Each time the RxDMA controller flushes one frame, this counter will plus 1. |

### 43.4.54. DMA receive state watchdog counter register (ENET_DMA_RSWDC)

Address offset: 0x1024
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

The watchdog counter value register for RS bit (ENET_DMA_STAT register) set after delay a configured time.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | WDCFRS[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | WDCFRS[7:0] | Watchdog counter for receive status (RS) bit<br>These bits are only valid when DINTC (RXDES1) is set. When DINTC=1 and a frame is received, the RS bit will be set delay a time of WDCFRS*256 HCLK after receiving complete. |

### 43.4.55. DMA current transmit descriptor address register (ENET_DMA_CTDADDR)

Address offset: 0x1048
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register points to the start descriptor address of the current transmit descriptor read by the TxDMA controller.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TDAP[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TDAP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TDAP[31:0] | Transmit descriptor address pointer bits<br>These bits are automatically updated by TxDMA controller during operation. |

### 43.4.56. DMA current receive descriptor address register (ENET_DMA_CRDADDR)

Address offset: 0x104C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register points to the start descriptor address of the current receive descriptor read by the RxDMA controller.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDAP[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| RDAP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RDAP[31:0] | Receive descriptor address pointer bits<br>These bits are automatically updated by RxDMA controller during operation. |

### 43.4.57. DMA current transmit buffer address register (ENET_DMA_CTBADDR)

Address offset: 0x1050
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register points to the current transmit buffer address being read by the TxDMA controller.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TBAP[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TBAP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | TBAP[31:0] | Transmit buffer address pointer bits<br>These bits are automatically updated by TxDMA controller during operation. |

### 43.4.58. DMA current receive buffer address register (ENET_DMA_CRBADDR)

Address offset: 0x1054
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit), half-word(16-bit) or word (32-bit).

This register points to the current receive buffer address being read by the RxDMA controller.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RBAP[31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RBAP[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RBAP[31:0] | Receive buffer address pointer bits<br>These bits are automatically updated by RxDMA controller during operation. |

# 44. Comparator (CMP)

## 44.1. Overview

The general purpose CMP can work either standalone (all terminal are available on I/Os) or together with the timers.

It can be used to wake up the MCU from low-power mode by an analog signal, provide a trigger source when an analog signal is in a certain condition, achieve some current control by working together with a PWM output of a timer and the DAC.

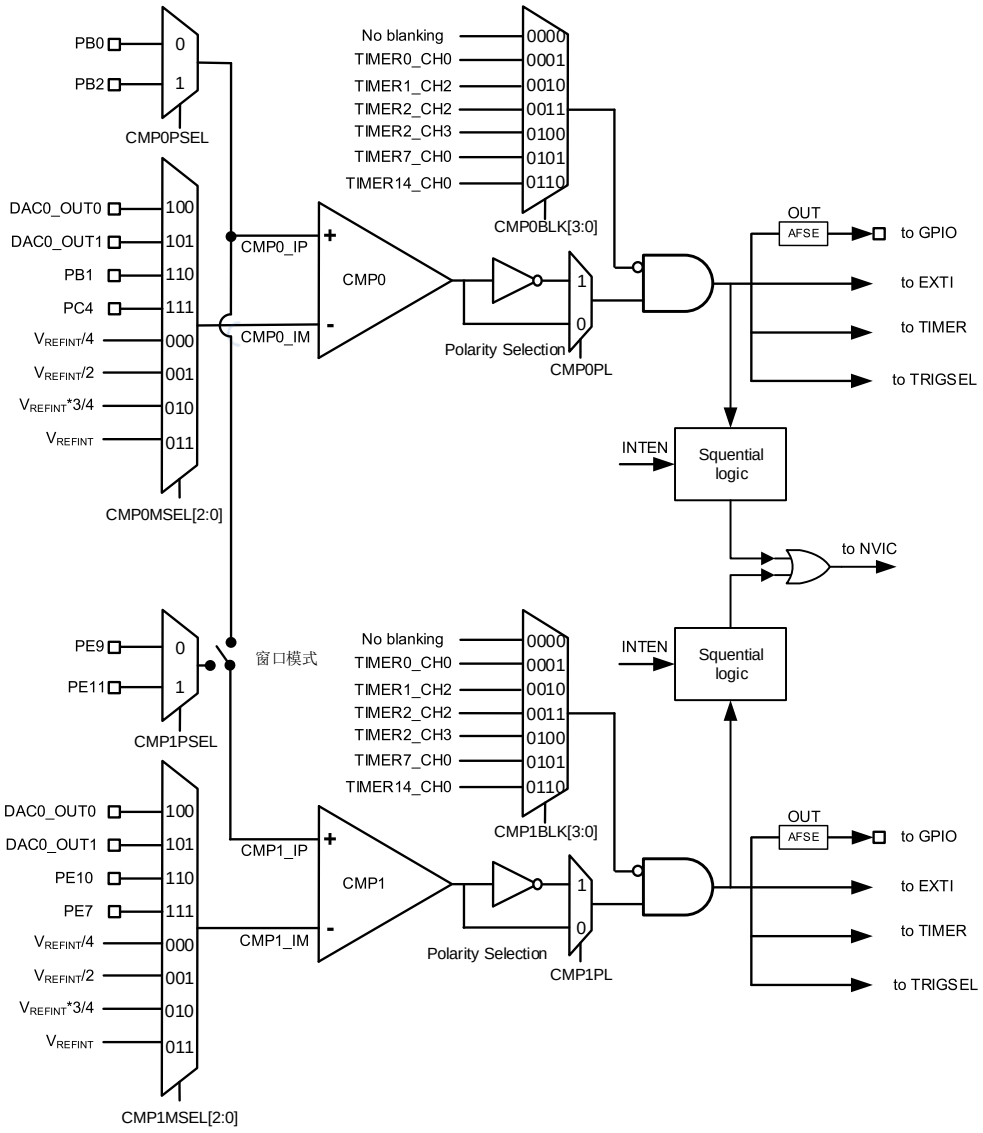
## 44.2. Characteristic

- Rail-to-rail comparators.
- Configurable hysteresis.
- Configurable speed and consumption.
- Configurable analog input source.
  - DAC output.
  - Multiplexed I / O pins.
  - The whole or sub-multiple values of internal reference voltage.
- Outputs with blanking source.
- Window comparator.
- Outputs to I / O.
- Outputs to timers for triggering.
- Outputs to EXTI.
- Outputs to NVIC.
- Outputs to TRIGSEL.

## 44.3. Function overview

The block diagram of CMP is shown below:

**Figure 44-1. CMP block diagram**




**Note:** $V_{REFINT}$ is 1.2V.

## 44.3.1. CMP clock

The clock of the CMP which is connected to APB bus, is synchronous with PCLK.

## 44.3.2. CMP I/O configuration

These I / Os must be configured in analog mode in the GPIOs registers before they are selected as CMP inputs.

The CMP output can be redirected internally and externally simultaneously.

Refer to pin definitions in datasheet, and the CMP output can be connected to the corresponding I/O port via the alternate function of the GPIO.

CMP output internally connect to the TIMER and the connections between them are as follows:

- CMP output to the TIMER input channel.
- CMP output to the TIMER break.

In order to work even in Deep-sleep mode, the polarity selection logic and the output redirection to the port work independently from PCLK.

*Table 44-1. CMP inputs and outputs summary* details the inputs and outputs of the CMP.

**Table 44-1. CMP inputs and outputs summary**

| | CMP0 | CMP1 |
|---|---|---|
| **CMP non inverting inputs connected to I/Os** | PB0<br>PB2 | PE9<br>PE11 |
| **CMP inverting inputs connected to I/Os** | PB1<br>PC4 | PE10<br>PE7 |
| **CMP inverting inputs connected to internal signals** | $V_{REFINT}$/4,<br>$V_{REFINT}$/2,<br>$V_{REFINT}$*3/4,<br>$V_{REFINT}$,<br>DAC0_OUT0,<br>DAC0_OUT1 | $V_{REFINT}$/4,<br>$V_{REFINT}$/2,<br>$V_{REFINT}$*3/4,<br>$V_{REFINT}$,<br>DAC0_OUT0,<br>DAC0_OUT1 |
| **CMP outputs connected to I/Os** | PC5 (AF13)<br>PE12 (AF13) | PE8 (AF13)<br>PE13 (AF13) |
| **CMP outputs connected to EXTI** | • | |
| **CMP outputs connected to TRIGSEL** | • | |
| **CMP outputs connected to NVIC** | • | |
| **CMP outputs connected to internal signals** | • | |
| **CMP outputs connected to TIMER break** | • | |
| **CMP_MUX_OUT (controlled by AFSE[x])** | PA6 (AF10)<br>PA8 (AF12)<br>PB12 (AF13)<br>PE6 (AF11)<br>PE15 (AF13)<br>PG2 (AF11)<br>PG3 (AF11)<br>PG4 (AF11)<br>PK0 (AF11)<br>PK1 (AF11)<br>PK2 (AF10) | |

**Note:**

1. Refer to the system configuration registers for details on CMP outputs connected to

internal signals.

2. Refer to the TIMER module for details on CMP outputs connected to TIMER break.

### 44.3.3. CMP operating mode

For a given application, there is a trade-off between the CMP power consumption versus propagation delay, which is adjusted by configuring bits CMPxM[1:0] in CMPx_CS register. The CMP works fastest with highest power consumption when CMPxM[1:0] = 2'b00, while works slowest with lowest power consumption when CMPxM [1:0]= 2'b11.

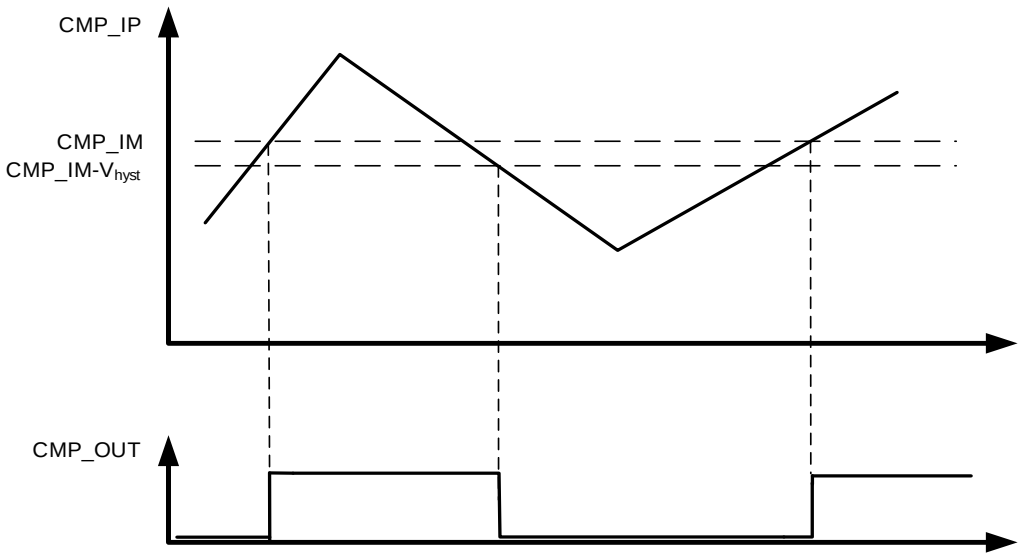
### 44.3.4. CMP Window mode

If the WNDEN bit in CMPx_CS register is set, comparator windows mode is enabled, input plus of comparator 1 is connected with input plus of comparator 0. If the minus input of CMP0 and CMP1 is connected to different voltage, the voltage range from lower threshold to upper threshold, is monitored by analyzing the comparator 0 and comparator 1 output.

### 44.3.5. CMP hysteresis

In order to avoid spurious output transitions that caused by the noise signal, a programmable hysteresis is designed to force the hysteresis value by configuring CMPx_CS register. This function could be shut down if it is unnecessary.

**Figure 44-2. CMP hysteresis**




### 44.3.6. CMP register write protection

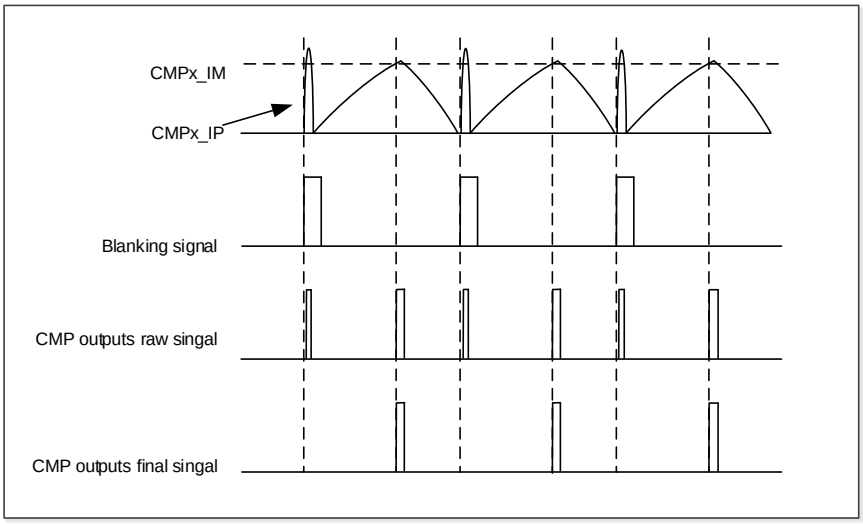
The CMP control and status register (CMPx_CS) and alternate select register (CMP_SR) can be protected from writing by setting CMPxLK bit to 1. The CMPx_CS register, including the CMPxLK bit will be read-only, and can only be reset by the MCU reset.

### 44.3.7. CMP output blanking

CMP output blanking function can be used to avoid interference of short pulses in the input signal to CMP output signal. If the CMPxBLK[2:0] bits in the CMPx_CS register are setting to an available value, the CMP output final signal is obtained by ANDing the complementary signal of the selected blanking signal with the raw output of the comparator. The blanking function can be used for false overcurrent detection in motor control applications.

*Figure 44-3. The CMP outputs signal blanking* shows the comparator output blank function.

**Figure 44-3. The CMP outputs signal blanking**




### 44.3.8. CMP voltage scaler function

The voltage scaler function can provide selectable 1 / 4, 1 / 2, 3 / 4 reference voltage for CMP input. It is controlled by CMPxSEN and CMPxBEN bits in CMPx control / status register. The CMPxSEN and CMPxBEN bits are used to enable the $V_{REFINT}$ voltage output and the divider circuit, respectively, to generate the selected voltage.

### 44.3.9. CMP interrupt

The CMP output is connected to the EXTI and the EXTI line is exclusive to CMP. With this function, CMP can generate either interrupt or event which could be used to exit from low-power mode.

The CMP also can generate an interrupt to NVIC. It is a sequential logic signal, so the PCLK is needed.

## 44.4. Register definition

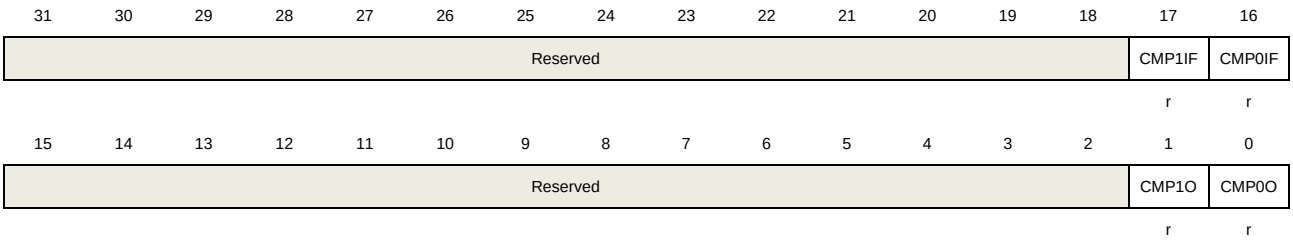
CMP base address: 0x5800 3800

### 44.4.1. CMP status register (CMP_STAT)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



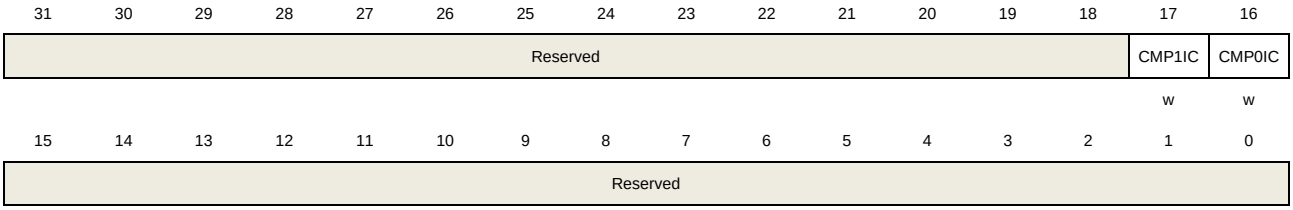

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | CMP1IF | CMP1 interrupt flag<br>0: No CMP1 output interrupt<br>1: CMP1 output interrupt<br>Set by hardware when the CMP1 output is set.<br>Cleared by software writing 1 to CMP1IC bit in the CMP_IFC register. |
| 16 | CMP0IF | CMP0 interrupt flag<br>0: No CMP0 output interrupt<br>1: CMP0 output interrupt<br>Set by hardware when the CMP0 output is set.<br>Cleared by software writing 1 to CMP0IC bit in the CMP_IFC register. |
| 15:2 | Reserved | Must be kept at reset value. |
| 1 | CMP1O | CMP1 output state<br>This bit is a copy of CMP1 output state, which is read only.<br>0: Non-inverting input below inverting input and the output is low<br>1: Non-inverting input above inverting input and the output is high |
| 0 | CMP0O | CMP0 output state<br>This bit is a copy of CMP0 output state, which is read only.<br>0: Non-inverting input below inverting input and the output is low<br>1: Non-inverting input above inverting input and the output is high |

### 44.4.2. CMP interrupt flag clear register (CMP_IFC)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



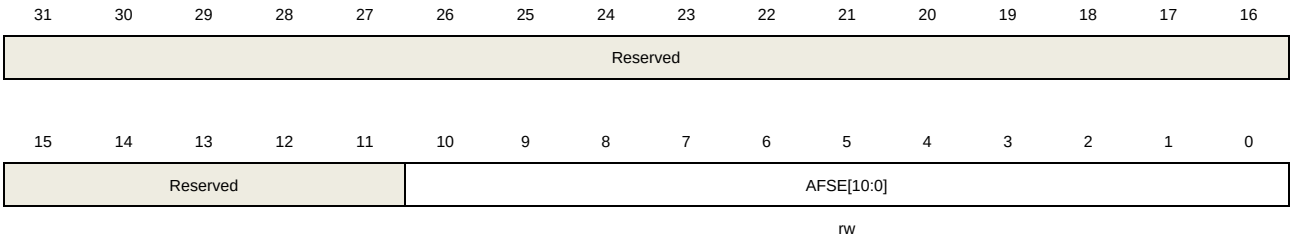

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | CMP1IC | CMP1 interrupt flag clear<br>0: Not clear CMP1 interrupt flag<br>1: Clear CMP1 interrupt flag |
| 16 | CMP0IC | CMP0 interrupt flag clear<br>0: Not clear CMP0 interrupt flag<br>1: Clear CMP0 interrupt flag |
| 15:0 | Reserved | Must be kept at reset value. |

### 44.4.3. CMP alternate select register (CMP_SR)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



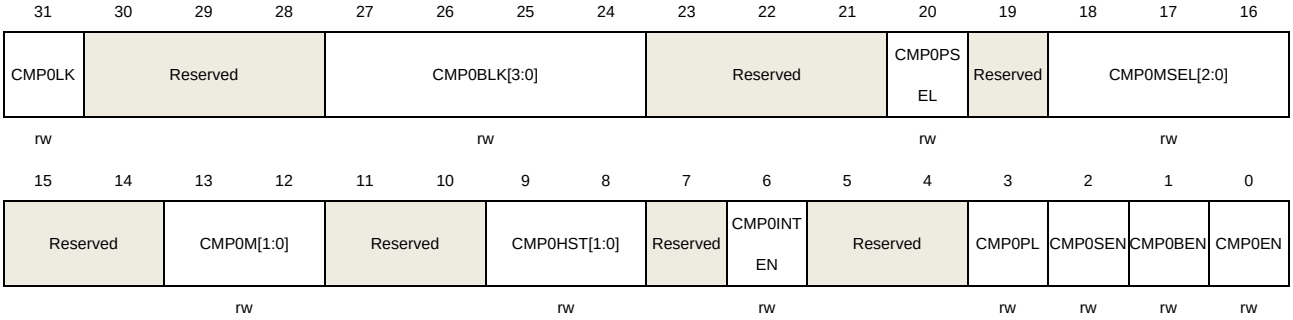

| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10:0 | AFSE[10:0] | CMP selects alternate output ports<br>For each bit, 0 is selected for CMP0_OUT as the alternate function with the corresponding GPIO, and 1 is selected for CMP1_OUT.<br>bit0: PA6<br>bit1: PA8 |

bit2: PB12
bit3: PE6
bit4: PE15
bit5: PG2
bit6: PG3
bit7: PG4
bit8: PK0
bit9: PK1
bit10: PK2

### 44.4.4. CMP0 control/status register (CMP0_CS)

Address offset: 0x0C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CMP0LK | CMP0 lock<br>This bit allows to have all control bits of CMP0 as read-only. It can only be set once by software and cleared by a system reset.<br>0: CMP0_CS bits are read-write<br>1: CMP0_CS and CMP_SR bits are read-only |
| 30:28 | Reserved | Must be kept at reset value. |
| 27:24 | CMP0BLK[3:0] | CMP0 output blanking source<br>This bit is used to select which timer output controls the comparator output blanking.<br>0000: No blanking<br>0001: Select TIMER0_CH0 output compare signal as blanking source<br>0010: Select TIMER1_CH2 output compare signal as blanking source<br>0011: Select TIMER2_CH2 output compare signal as blanking source<br>0100: Select TIMER2_CH3 output compare signal as blanking source<br>0101: Select TIMER7_CH0 output compare signal as blanking source<br>0110: Select TIMER14_CH0 output compare signal as blanking source<br>All other values: reserved. |

| | | |
|---|---|---|
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | CMP0PSEL | CMP0_IP input selection<br>This bit is used to select the source connected to the CMP0_IP input of the CMP0.<br>0: PB0<br>1: PB2 |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | CMP0MSEL[2:0] | CMP0_IM internal input selection<br>These bits are used to select the internal source connected to the CMP0_IM input of the CMP0.<br>000: $V_{REFINT}$ / 4<br>001: $V_{REFINT}$ / 2<br>010: $V_{REFINT}$*3 / 4<br>011: $V_{REFINT}$<br>100: DAC0_OUT0<br>101: DAC0_OUT1<br>110: PB1<br>111: PC4 |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:12 | CMP0M[1:0] | CMP0 mode<br>These bits are used to control the operating mode of the CMP0 adjust the speed / consumption.<br>00: High speed / full power<br>01 / 10: Medium speed / medium power<br>11: Very-low speed / ultra-low power |
| 11:10 | Reserved | Must be kept at reset value. |
| 9:8 | CMP0HST[1:0] | CMP0 hysteresis<br>These bits are used to control the hysteresis level.<br>00: No hysteresis<br>01: Low hysteresis<br>10: Medium hysteresis<br>11: High hysteresis |
| 7 | Reserved | Must be kept at reset value. |
| 6 | CMP0INTEN | CMP0 interrupt enable<br>0: Disabled<br>1: Enabled |
| 5:4 | Reserved | Must be kept at reset value. |
| 3 | CMP0PL | Polarity of CMP0 output<br>This bit is used to select the polarity of CMP0 output.<br>0 : Output is not inverted |

| | | |
|---|---|---|
| | | 1 : Output is inverted |
| 2 | CMP0SEN | Voltage scaler enable bit<br>This bit is set and cleared by software. This bit enables the outputs of the VREFINT divider, which is treated as the minus input of the comparator.<br>0: Disable bandgap scaler in case that CMP1SEN bit of CMP1_CS is also reset<br>1: Enable bandgap scaler enable |
| 1 | CMP0BEN | Scaler bridge enable bit<br>0: Disable scaler resistor bridge in case that CMP1BEN bit of CMP01_CS is also reset<br>1: Enable scaler resistor bridge |
| 0 | CMP0EN | CMP0 enable<br>0: CMP0 disabled<br>1: CMP0 enabled |

### 44.4.5. CMP1 control/status register (CMP1_CS)

Address offset: 0x10
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMP1LK | Reserved | | | CMP1BLK[3:0] | | | | Reserved | | | CMP1PSEL | Reserved | CMP1MSEL[2:0] | | |
| rw | | | | rw | | | | | | | rw | | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CMP1M[1:0] | | Reserved | | CMP1HST[1:0] | | Reserved | CMP1INTEN | Reserved | WNDEN | CMP1PL | CMP1SEN | CMP1BEN | CMP1EN |
| | | rw | | | | rw | | | rw | | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CMP1LK | CMP1 lock<br>This bit allows to have all control bits of CMP1 as read-only. It can only be set once by software and cleared by a system reset.<br>0: CMP1_CS bits are read-write<br>1: CMP1_CS and CMP_SR bits are read-only |
| 30:28 | Reserved | Must be kept at reset value. |
| 27:24 | CMP1BLK[3:0] | CMP1 output blanking source<br>This bit is used to select which timer output controls the comparator output blanking.<br>0000: No blanking<br>0001: Select TIMER0_CH0 output compare signal as blanking source<br>0010: Select TIMER1_CH2 output compare signal as blanking source<br>0011: Select TIMER2_CH2 output compare signal as blanking source |

| | | |
|---|---|---|
| | | 0100: Select TIMER2_CH3 output compare signal as blanking source<br>0101: Select TIMER7_CH0 output compare signal as blanking source<br>0110: Select TIMER14_CH0 output compare signal as blanking source<br>All other values: reserved. |
| 23:21 | Reserved | Must be kept at reset value. |
| 20 | CMP1PSEL | CMP1_IP input selection<br>This bit is used to select the source connected to the CMP1_IP input of the CMP1.<br>0: PE9<br>1: PE11 |
| 19 | Reserved | Must be kept at reset value. |
| 18:16 | CMP1MSEL[2:0] | CMP1_IM internal input selection<br>These bits are used to select the internal source connected to the CMP1_IM input of the CMP1.<br>000: $V_{REFINT}$ / 4<br>001: $V_{REFINT}$ / 2<br>010: $V_{REFINT}$*3 / 4<br>011: $V_{REFINT}$<br>100: DAC0_OUT0<br>101: DAC0_OUT1<br>110: PE10<br>111: PE7 |
| 15:14 | Reserved | Must be kept at reset value. |
| 13:12 | CMP1M[1:0] | CMP1 mode<br>These bits are used to control the operating mode of the CMP1 adjust the speed / consumption.<br>00: High speed / full power<br>01 / 10: Medium speed / medium power<br>11: Very-low speed / ultra-low power |
| 11:10 | Reserved | Must be kept at reset value. |
| 9:8 | CMP1HST[1:0] | CMP1 hysteresis<br>These bits are used to control the hysteresis level.<br>00: No hysteresis<br>01: Low hysteresis<br>10: Medium hysteresis<br>11: High hysteresis |
| 7 | Reserved | Must be kept at reset value. |
| 6 | CMP1INTEN | CMP1 interrupt enable<br>0: Disabled<br>1: Enabled |

| | | |
|---|---|---|
| 5 | Reserved | Must be kept at reset value. |
| 4 | WNDEN | Window mode enable<br>This bit is used to select CMP1_IP source.<br>0: CMP1_IP is connected to CMP1 non-inverting input<br>1: CMP1_IP is connected to CMP0_IP |
| 3 | CMP1PL | Polarity of CMP1 output<br>This bit is used to select the polarity of CMP1 output.<br>0 : Output is not inverted<br>1 : Output is inverted |
| 2 | CMP1SEN | Voltage scaler enable bit<br>This bit is set and cleared by software. This bit enables the outputs of the VREFINT divider, which is treated as the minus input of the comparator.<br>0: Disable bandgap scaler disable in case that CMP0SEN bit of CMP0_CS is also reset<br>1: Enable bandgap scaler enable |
| 1 | CMP1BEN | Scaler bridge enable bit<br>0: Disable scaler resistor bridge in case that CMP0BEN bit of CMP0_CS is also reset<br>1: Enable scaler resistor bridge |
| 0 | CMP1EN | CMP1 enable<br>0: CMP1 disabled<br>1: CMP1 enabled |

# 45. High-Performance Digital Filter (HPDF)

## 45.1. Overview

A high performance digital filter module (HPDF) for external sigma delta (Σ-Δ) modulator is integrated in GD32H73x_75x. HPDF supports SPI interface and Manchester-coded single-wire interface. The external sigma delta modulator can be connected with MCU by the serial interface, and the serial data stream output by sigma delta modulator can be filtered. In addition, HPDF also supports the parallel data stream input, which can be selected from internal ADC peripherals or from MCU memory.

## 45.2. Characteristics

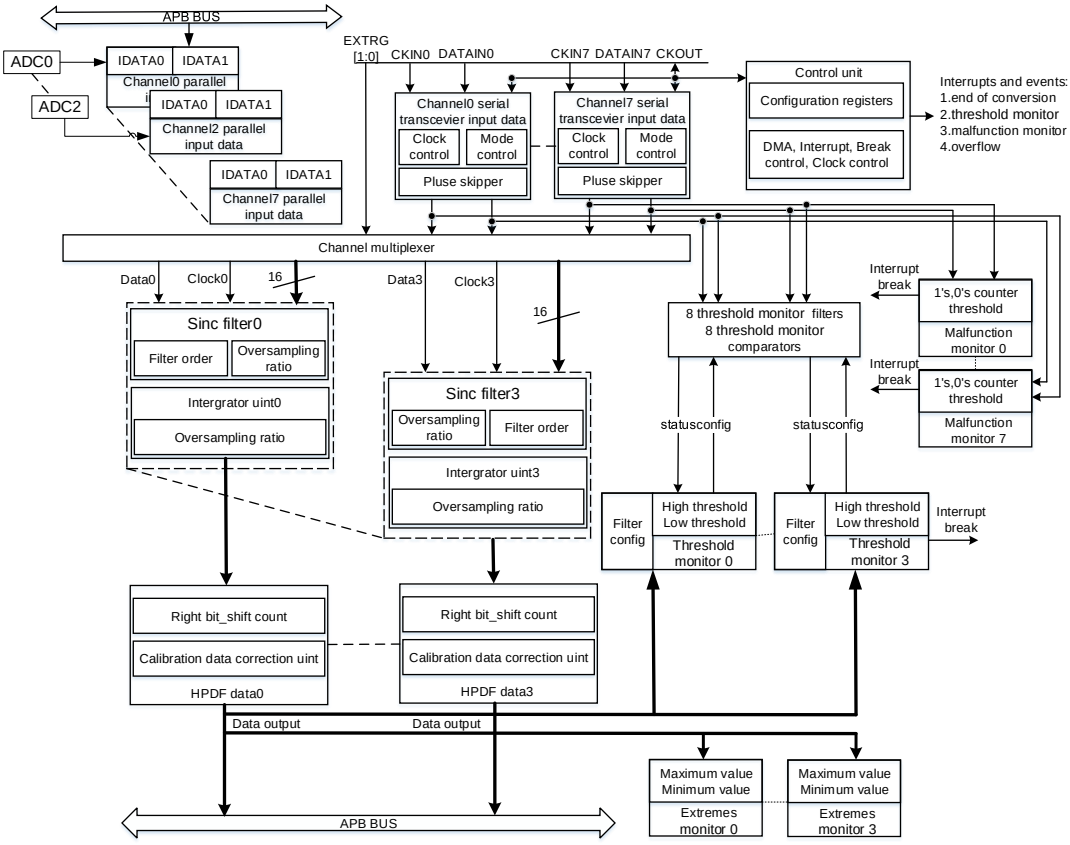
- 8 multiplex digital serial input channels
  - configurable SPI and Manchester interfaces
- 8 internal digital parallel input channels
  - input with up to 16-bit resolution
  - internal source: ADC data or memory (CPU/DMA write) data stream
- Configurable Sinc filter and integrator
  - the order and oversampling rate (decimation rate) of Sinc filter can be configured;
  - sampling rate of configurable integrator
- Threshold monitor function
  - independent Sinc filter, configurable order and oversampling rate (decimation rate)
  - configurable data input source: serial channel input data or HPDF output data
- Malfunction monitor function
  - A counter with 8 bits is used to monitor the continuous 0 or 1 in the serial channel input data stream
- Extreme monitor function
  - store minimum and maximum values of output data values of HPDF
- Up to 24-bit output data resolution
- Clock signal can be provided to external sigma delta modulator
  - provide configurable clock signal by the CKOUT pin
- Flexible conversion configuration function
  - the conversion channel is divided into regular group and inserted group
  - support multiple conversion modes and startup modes
- HPDF output data is in signed format

# 45.3. Function overview

## 45.3.1. HPDF Block Diagram

The structural block diagram of HPDF is shown in ***Figure 45-1. HPDF block diagram***.

**Figure 45-1. HPDF block diagram**




The HPDF interface communicates with the external Σ-Δ modulator by the pins and internal signal in ***Table 45-1. HPDF pins definition***.

**Table 45-1. HPDF pins definition**

| PINs | Type | Description |
|---|---|---|
| EXTRG[1:0] | External trigger input | Input pin of external trigger signal source, the trigger signal sources are EXTI11 and EXTI15, which are used as the trigger signal of inserted group HPDF_ ITRG[24] and HPDF_ITRG[25] |
| CKOUT | Clock out | The clock output signal of HPDF module, provides clock signal to external Σ-Δ modulator. |
| CKINx | Clock input | External Σ-Δ modulator provides clock signal to serial interface. |

| PINs | Type | Description |
|---|---|---|
| DATAINx | Data input | The external Σ-Δ modulator transmits 1 bit data stream to the serial channel by this pin. |

**Table 45-2. HPDF internal signal**

| Break name | Break destination |
|---|---|
| HPDF_BREAK[0] | TIMER0 break0 / TIMER14 break0 / TIMER41 break0 |
| HPDF_BREAK[1] | TIMER0 break1 / TIMER15 break0 / TIMER42 break0 |
| HPDF_BREAK[2] | TIMER7 break0 / TIMER16 break0 / TIMER43 break0 |
| HPDF_BREAK[3] | TIMER7 break1 / TIMER40 break0 / TIMER44 break0 |

### 45.3.2. HPDF on-off control

When the HPDF module is started normally, the HPDF module can be enabled globally by setting HPDFEN to 1 in the HPDF_CH0CTL register. Then set the CHEN bit in HPDF_CHxCTL and the FLTEN bit in HPDF_FLTyCTL0 to 1 to enable the input channel and channel digital filter respectively. In addition, as long as the input channel is enabled, the input channel will immediately start receiving serial data.

HPDF can enter stop mode by clearing FLTEN during operation. After entering stop mode, the ongoing conversion tasks of the HPDF module will immediately stop, and the configuration of the registers remains unchanged (except for the HPDF_FLTySTAT and HPDF_FLTyTMSTAT registers are reset).

In stop mode, the HPDF system clock will automatically stop. The HPDFEN bit must be cleared before the system clock is stopped to enter stop mode.

#### Low power mode

HPDF module optimizes the reduction of power consumption. In the normal working mode, the filter and integrator will automatically enter the idle state to achieve the purpose of reducing power consumption when there is no conversion task.

### 45.3.3. HPDF clock

The clock of HPDF includes the system clock and the serial clock. The system clock is used to drive the internal modules, and the serial clock used by the serial interface.

#### System clock

The system clock $f_{HPDFCLK}$ of HPDF is used to drive channel transceiver, digital filter, integrator, threshold monitor, malfunction monitor, extremum detector and control module. The HPDF system clock source can be configured by the HPDFSEL bit in the RCU_CFG1 register of the RCU chapter.

### Serial input clock

The serial interface of HPDF can receive clock signal from external sigma delta modulator by CKINx pin, so as to receive the serial data stream from sigma delta modulator.

Using external input clock in serial interface is limited by clock frequency. If the standard SPI interface is used, the system clock $f_{HPDFCLK} \geq 4f_{CKIN}$. If the Manchester coding interface is used, the system clock $f_{HPDFCLK} \geq 6f_{CKIN}$ is required.

### Serial output clock

HPDF supports the function of outputting serial clock, which can drive sigma delta modulator connected with it. The source of the serial output clock can be selected by CKOUTSEL bit in HPDF_CH0CTL register. When CKOUTSEL=0, the serial output clock source is the HPDF system clock. When CKOUTSEL=1, the serial output clock source is the audio clock. And the configuration of the audio clock can refer to the SAI0SEL[2:0] bit field in the configuration register 2 of the RCU chapter.

After the serial output clock source is determined, the output clock frequency division can be controlled by configuring the CKOUTDIV [7:0] bit field in the HPDF_CH0CTL register. When CKOUTDIV[7:0] ≠ 0, the value of the serial output clock divider is CKOUTDIV[7:0]+1. When CKOUTDIV[7:0] = 0, the serial output clock is disabled and the pin of CKOUT remains low.

In addition, after clearing HPDFEN, the signal of serial output clock can also be stopped. When the serial output clock source is the system clock (CKOUTSEL = 0), if clear HPDFEN, the serial output clock stopped after 4 system clocks. When the serial output clock source is the audio clock (CKOUTSEL = 1), if clear HPDFEN, the serial output clock stopped after one system clock and three audio clocks.

The serial output clock source can only be modified when HPDFEN = 0. In order to avoid the burr signal on the pin of CKOUT, the software can only modify the value of the CKOUTSEL bit in the HPDF_CH0CTL register after the serial output clock stopped.

The frequency range of the serial output clock is 0-20MHz.

## 45.3.4. Multiplex serial data channel

HPDF has eight multiplexing serial data channels, which support SPI code and Manchester code. The interface type supported can be selected for the current channel by configuring the SITYP[1:0] bit field in the HPDF_CHxCTL register.

### SPI interface

Under the standard SPI interface, sigma delta modulator sends 1-bit data stream to the serial channel by the pin of DATAINx. The clock signal between HPDF and sigma delta modulator can be output by CKOUT pin or input by CKINx pin.

The data sampling point in SPI communication is determined by the SITYP[1:0] bit field and

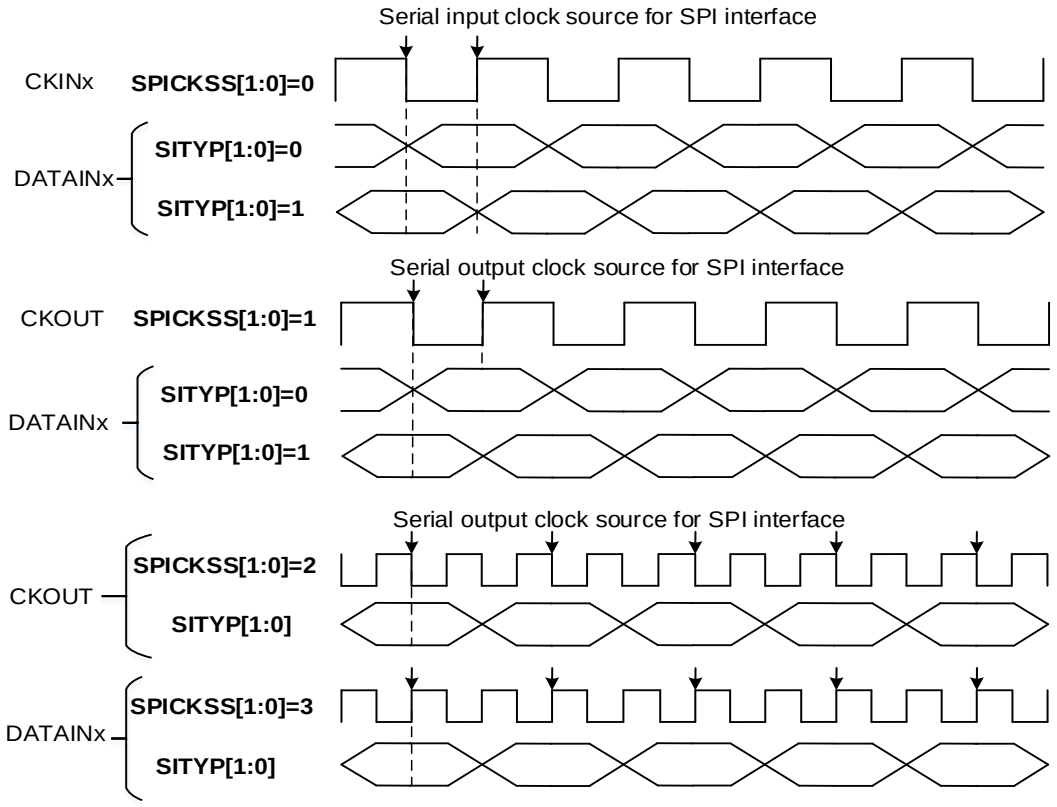
SPICKSS[1:0] bit field in HPDF_CHxCTL register. The data sampling points in SPI communication are shown in the table.

**Table 45-3. SPI interface clock configuration**

| SPICKSS[1:0] | Clock source | SITYP[1:0] | Sampling point | Description |
|---|---|---|---|---|
| 00 | CKINx signal | 00 | rising edge | Data is sampled at the rising edge of the external serial input clock signal |
| | | 01 | falling edge | Data is sampled at the falling edge of the external serial input clock signal |
| 01 | CKOUT signal | 00 | rising edge | The data is sampled at the rising edge of the internal serial output clock signal |
| | | 01 | falling edge | The data is sampled at the falling edge of the internal serial output clock signal |
| 10 | CKOUT/2 signal (Generated at the rising edge of CKOUT) | xx | Falling edge of each second CKOUT signal | The external sigma delta modulator divides the CKOUT signal into 2 frequencies to generate the serial input communication clock. The data is sampled at the falling edge of every second CKOUT. |
| 11 | CKOUT/2 signal (Generated at the falling edge of CKOUT) | xx | Rising edge of each second CKOUT signal | The external sigma delta modulator divides the CKOUT signal into 2 frequencies to generate the serial input communication clock. The data is sampled at the rising edge of every second CKOUT. |

According to ***Table 45-3. SPI interface clock configuration***, the sequence diagram of SPI data transmission is shown in the figure below.

**Figure 45-2. The sequence diagram of SPI data transmission**




**Note:** if SPI data interface is adopted, the frequency range of clock source is 0-20MHz and less than $f_{HPDFCLK}/4$.

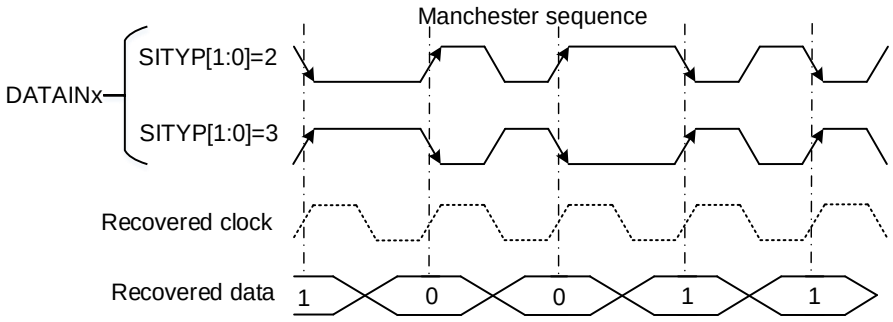
### Manchester interface

HPDF has eight multiplexing serial data channels using Manchester encoding format. Two encoding formats can be configured by SITYP[1:0] bit field in HPDF_CHxCTL:

1. When SITYP[1:0] = 2, Manchester code: rising edge = logic 0, falling edge = logic 1.
2. When SITYP[1:0] = 3, Manchester code: rising edge = logic 1, falling edge = logic 0.

When Manchester code is used, the data stream between the external sigma delta modulator and HPDF is only transmitted by the DATAINx pin. After the HPDF module Manchester decoding, the clock signal and data are recovered from the serial data stream. The recovered clock signal frequency must be between 0-10MHz and less than $f_{HPDFCLK}/6$. The timing chart of Manchester data transmission is shown in the figure below.

**Figure 45-3. The sequence diagram of Manchester data transmission**




In order to receive and decode Manchester data correctly, configure the CKOUTDIV[7:0] frequency divider according to the expected flow rate of Manchester data. The value of CKOUTDIV[7:0] is calculated with reference to the following format:

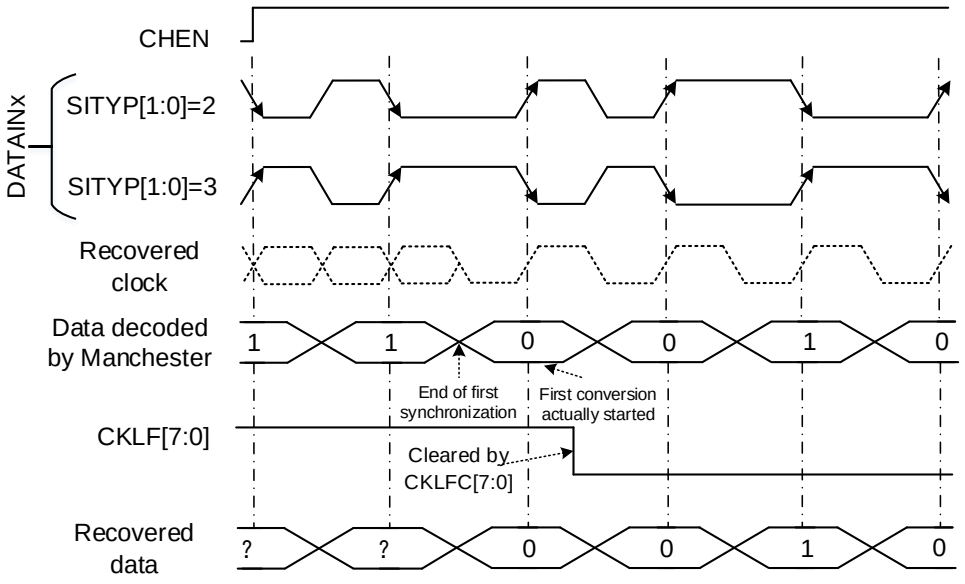
$$((CKOUTDIV+1)xT_{SYSCLK})<T_{Manchester_clock}<(2xCKOUTDIV\times T_{SYSCLK}) \quad (45\text{-}1)$$

## Serial communication coding synchronization

After the serial channel is enabled, the data can only be received correctly after successful synchronization. The synchronization of SPI code occurs after the first detection of clock input signal by SPI data stream. If the channel uses Manchester coding, the first synchronization occurs when the channel receives data stream changes from 1-0 or 0-1.

Before the transceiver of the serial channel synchronizes, the clock loss flag bit of channel is set to 1. After successful synchronization, the clock loss flag bit can be cleared by CKLFC[7:0]. When the transceiver of the serial channel is not synchronized, the clock loss flag bit cannot be cleared by the CKLFC[7:0]. Therefore, it is possible to determine whether the serial channel is successfully synchronized by querying the CKLF[7:0] bit circularly. The following figure shows the timing chart of the first synchronization of Manchester code.

**Figure 45-4. Manchester synchronous sequence diagram**



### External serial clock frequency measurement

The measuring of a channel serial clock input frequency provides a real data rate from an external ΣΔ modulator, which is important for application purposes.

An external serial clock input frequency can be measured by a timer counting HPDF clocks ($f_{HPDFCLK}$) during one conversion duration. The counting starts at the first input data clock after a conversion trigger (regular or injected) and finishes by last input data clock before conversion ends (end of conversion flag is set). Each conversion duration (time between first serial sample and last serial sample) is updated in counter CNVCNT[27:0] in register HPDF_FLTxCT when the conversion finishes (ICEF=1 or RCEF=1). The user can then compute the data rate according to the digital filter settings (SFO, SFOR, IOR, FAST). The external serial frequency measurement is stopped only if the filter is bypassed (SFOR=0, only integrator is active, CNVCNT[27:0]=0 in HPDF_FLTxCT register).

In case of parallel data input the measured frequency is the average input data rate during one conversion.

Note:When conversion is interrupted (by disabling/enabling the selected channel) the interruption time is also counted in CNVCNT[27:0]. Therefore it is recommended to not interrupt the conversion for correct conversion duration result.

Conversion times:

injected conversion or regular conversion with FAST = 0 (or first conversion if FAST=1):

for Sincx filters:

$$T = CNVCNT / f_{HPDFCLK} = [SFOR * (IOR-1 +SFO) + SFO] / f_{CKIN}$$

for FastSinc filter:

$$T = CNVCNT / f_{HPDFCLK} = [SFOR * (IOR-1 +4) + 2] / f_{CKIN})$$

regular conversion with FAST = 1 (except first conversion):

for Sincx and FastSinc filters:

$$T = CNVCNT / f_{HPDFCLK} = [SFOR * IOR] / f_{CKIN})$$

in case if FOSR = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

$$T = IOR / f_{CKIN} \text{ (but CNVCNT=0)}$$

where:

- $f_{CKIN}$ is the channel input clock frequency (on given channel CKINx pin) or input data rate (in case of parallel data input)
- SFOR is the filter oversampling ratio: SFOR = SFOR[9:0]+1 (see HPDF_FLTxSFCFG register)
- IOR is the integrator oversampling ratio: IOR = IOR[7:0]+1 (see HPDF_FLTxSFCFG register)
- SFO is the filter order: SFO = SFO[2:0] (see HPDF_FLTxSFCFG register)

### Clock loss detection

Clock loss detection is to detect whether the channel serial input clock (CKINx signal) is lost, so as to ensure whether there is any error in the data of serial channel conversion (or threshold monitor and malfunction monitor). If a clock signal loss event occurs, the given data should be discarded. When using the clock loss detection function, you must configure the ckout signal source as the system clock.

The clock loss detection function can be enabled or disabled by the CKLEN bit in HPDF_CHxCTL register. When the enable clock loss detection function and the clock loss interrupt CKLIE occur, if a clock loss event occurs, the clock loss flag bit (CKLF) will be set to 1 and a clock loss interrupt will be generated. The corresponding interrupt flag bit can be cleared by setting CKLFC[7:0] bit field.

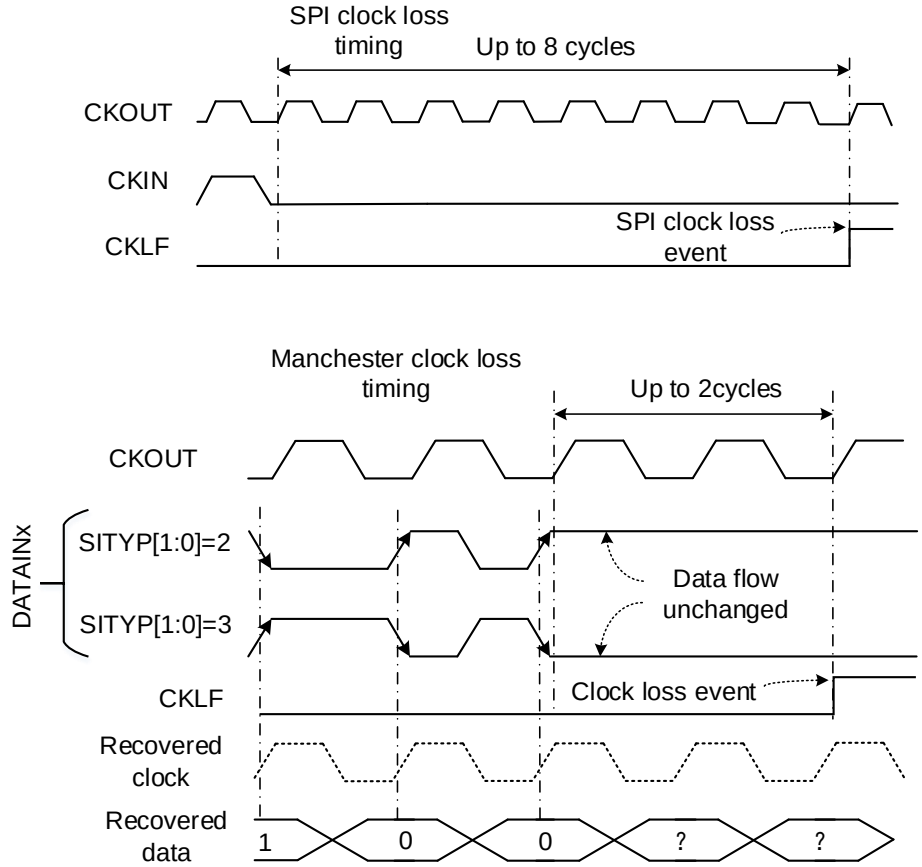
When the transceiver of the serial interface has not been synchronized, the clock loss flag bit is set to 1 and cannot be cleared by the corresponding CKLFC[7:0]. Therefore, the correct steps to use the clock loss function are as follows:

1. Enable the channel CHEN = 1.
2. Check the clock loss flag cyclically and write 1 to the CKLFC of the channel. When it is confirmed that the CKLF bit is cleared, the serial channel transceiver synchronization is successful.
3. Enable clock loss detection function CKLEN = 1. To detect possible clock loss, enable clock loss interrupt CKLIE = 1.

If the SPI interface is used in the serial channel, the external serial input clock (CKINx signal) is compared with the serial output clock (ckout signal) when the clock loss detection function is used. The external serial input clock signal must be inverted at least once every 8 CKOUT signal cycles, otherwise a clock loss event will occur.

If the serial channel uses the Manchester interface, the clock loss detection starts after the first successful synchronization of the Manchester code, and the external serial input data (DATAINx signal) is compared with the serial output clock (CKOUT signal). The serial input data must change every 2 ckout signal cycles, otherwise clock loss event will be generated. The timing of clock loss is shown in the figure below.

**Figure 45-5. Clock loss detection timing diagram**




**Note:** the maximum rate of Manchester encoded data stream must be less than the clock output CKOUT signal.

### Channel pin redirection

Channel pin redirection means that the pins of serial channel 0 can be configured as the pins of channel 1, that is, channel 0 can read information from the DATAIN1 and CKIN1 pins. Pin redirection is used to sampling audio data of PDM microphone. The audio signal of PDM microphone includes data and clock signal. The data is divided into left/right channel data. The left channel data is sampled at the rising edge of clock signal, and the right channel data is sampled at the falling edge of clock signal.

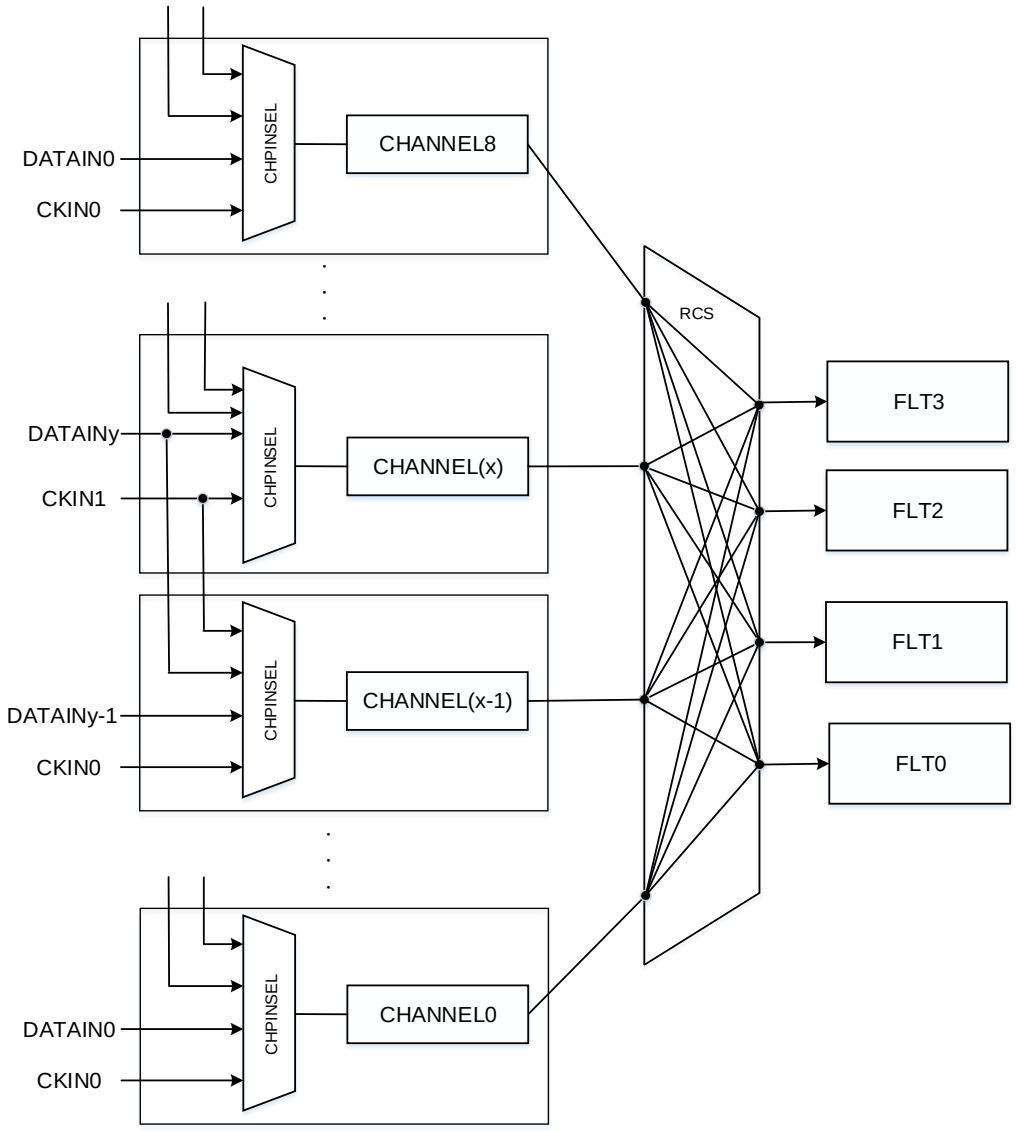
When PDM microphone data stream is input into serial channel, its configuration process is as follows:

1. Select the HPDF serial channel 1 of PDM microphone data stream input.
2. Write 0 to CHPINSEL bit of channel 1 in HPDF_CHxCTL register, and input pin of channel 1 is own pin, DATAINx and CKINx. When SITYP[1:0] = 2b’00, the serial data stream is sampled at the rising edge of the clock signal, that is, the input of channel 1 is the left channel data.
3. Set the CHPINSEL to 1 in channel 0, and the DATAINx and CKINx pins will be used for channel 0. When SITYP[1:0] = 2b’01, the serial data stream is sampled at the falling edge of the clock signal, that is, the input of channel 0 is the right channel data.

4. Configure channelx with corresponding filters to filter the left and right channel data of PDM microphone.

The channel pin redirection diagram of HPDF module is shown in ***Figure 45-6. Channel pins redirection.***

**Figure 45-6. Channel pins redirection**




## Pulses skipper

Pulse skipper refers to that the serial input data stream enters the filter after skipping a specified number of clock pulses, so as to discard a certain number of bit bits. This operation will cause the final output sample (and the next sample) from the filter to be calculated from the subsequent input data compared to the data stream that was not skipped.

The number of pulses to be skipped is determined by the PLSK[5:0] bit field in the HPDF_CHxPS register. Write the value to PLSK[5:0] bit field, and the specified channel will start to perform the pulse skipper function. Read the of PLSK[5:0], indicating the number of

remaining pulse skipper not executed. For a single write operation of PLSK[5:0], the maximum number of pulse skipper executed is 63. More pulse skipper can be obtained by writing to PLSK[5:0] bit field several times.

#### Serial input interface configuration

The configuration steps of serial input interface of HPDF module are as follows:

1. Configure clock output prescaler: by configuring the CKOUTDIV[7:0] bit field in the HPDF_CH0CTL register, the coefficient of prescaler is CKOUTDIV[7:0] + 1.
2. Configure the serial interface type and input clock phase: configure the serial interface type as SPI code or Manchester code, and determine the clock input sampling edge by the SITYP[1:0] bit field in HPDF_CHxCTL register.
3. Configure input clock source: select the clock source of serial interface as serial input clock or serial output clock by configuring SPICKSS[1:0] in HPDF_CHxCTL register.
4. Configure data offset correction and shift right bits: DTRS[4:0] defines the bits of the final data shift right in HPDF_CHxCFG register. After data shift, perform offset calibration defined by CALOFF [23:0] bit field.
5. Enable short circuit detection and clock loss detection function: enable short circuit detection and clock loss detection function by setting MMEN and CKLEN to 1.
6. Set the threshold monitor filter and malfunction monitor: the filter parameters of the threshold monitor, the malfunction signal allocation of the malfunction monitor and the counter threshold are all configured by the HPDF_CHxCFG1 register.

### 45.3.5. Parallel data input

HPDF module can select parallel data as the data input source of the channel. The CMSD[1:0] bit field in HPDF_CHxCTL is configured to determine whether the channel data input source is from serial data or parallel data. Each channel provides a 32-bit parallel data input register (HPDF_CHxPDI), which can write two 16 bit parallel data by CPU/DMA. The register has two 16-bit data in signed format.

#### Input from internal ADC

For parallel ADC data input (CMSD[1:0]), The ADC[x] result is assigned to channel x input. The end of conversion event from ADC[x] causes the data of channel x to be updated (the parallel data from ADC[x] is used as the next sampling for the digital filter). When the end of conversion event occurs, data from ADC[x] is written to the HPDF_CHxPDI register (DATAIN0[15:0]).

Data packing mode setting (DPM[1:0] in register HPDF_CHxCTL) has no effect on ADC data input.

#### CPU / DMA write parallel data

There are two ways to write parallel data: CPU direct write and DMA write. When using DMA to write parallel data, DMA should be configured as memory to memory mode, and its target

address is the address of HPDF_CHxPDI.

**Note:** DMA writing parallel data is different from DMA reading final conversion data from HPDF module. The latter needs to be configured in peripheral to memory mode.

### Parallel data packed mode

The data stored in HPDF_CHxPDI register will be processed by channel filter. There are three modes of parallel data stored in the HPDF_CHxPDI register.In different data packed modes, the number of filter samples allowed to load depends on the value of DPM [1:0] bit field in the HPDF_CHxCTL register. The different data encapsulation modes are as follows:

1. Standard mode (DPM[1:0] = 2'b00):

   In this mode, the upper 16 bits in the HPDF_CHxPDI register are write protected, and the 16-bit data written by CPU/DMA is stored in the low 16 bit DATAIN0[15:0] bit field. CPU/DMA is configured as a 16-bit access mode. When writing 16-bit data once, the channel filter must perform an input sampling to clear the HPDF_CHxPDI register.

2. Interleaving mode (DPM[1:0] = 2'b01):

   In this mode, the CPU/DMA is configured as a 32-bit access mode, and the data is stored in the DATAIN0[15:0] bit domain of the lower 16 bits and the DATAIN1[15:0] bit domain of the higher 16 bits. When writing 32-bit data once, the channel filter must perform two input samples to clear the HPDF_CHxPDI register. The channel filter samples the DATAIN0[15:0] bit domain for the first time and the DATAIN1[15:0] bit domain for the second time.

3. Dual channel mode (DPM[1:0] = 2'b10):

   In this mode, the CPU/DMA is configured as a 32-bit access mode, and the data is stored in the DATAIN0[15:0] bit domain of the lower 16 bits and the DATAIN1[15:0] bit domain of the higher 16 bits. The data in the DATAIN0[15:0] bit field is used for the current channel x, and the data in the DATAIN1[15:0] bit field is automatically copied to the lower 16 bits of the parallel data input register of the channel x+1, and the data is used for the channel x+1. CPU/DMA writes data once, digital filter performs two sampling, the first is channel x sampling, the second is channel x+1 sampling.

In HPDF module, only even channel (channel0) supports dual channel mode. If odd channel (channel1) is configured as dual channel mode, the parallel data input register HPDF_CHxPDI of this channel is write protected. If channel x is even and configured as dual channel mode, odd channel x+1 must be configured as standard mode.

The operation mode of HPDF_CHxPDI register is as follows:

**Table 45-4. Parallel data packed mode**

| Channel | Packed mode | | | | | |
|---|---|---|---|---|---|---|
| | Standard mode | | Interleaving mode | | Dual channel mode | |
| | DATAIN1 | DATAIN0 | DATAIN1 | DATAIN0 | DATAIN1 | DATAIN0 |
| Channel0 | Write protect | CH0 sampling | CH0 second sampling | CH0 first sampling | CH1 sampling | CH0 sampling |
| Channel1 | Write protect | CH1 sampling | CH1 second sampling | CH1 first sampling | Write protect | CH1 sampling |
| Channel2 | Write protect | CH2 sampling | CH2 second sampling | CH2 first sampling | CH3 sampling | CH2 sampling |
| Channel3 | Write protect | CH3 sampling | CH3 second sampling | CH3 first sampling | Write protect | CH3 sampling |
| Channel4 | Write protect | CH4 sampling | CH4 second sampling | CH4 first sampling | CH5 sampling | CH4 sampling |
| Channel5 | Write protect | CH5 sampling | CH5 second sampling | CH5 first sampling | Write protect | CH5 sampling |
| Channel6 | Write protect | CH6 sampling | CH6 second sampling | CH6 first sampling | CH7 sampling | CH6 sampling |
| Channel7 | Write protect | CH7 sampling | CH7 second sampling | CH7 first sampling | Write protect | CH7 sampling |

CPU/DMA should write to HPDF_CHxPDI register after the channel is enabled, because after the channel is enabled, the channel conversion will be started, and the data in HPDF_CHxPDI register will be discarded before the channel conversion is started.

### 45.3.6. Regular group conversion

HPDF module has 8 multiplexing channels, which can be used for regular group conversion or inserted group conversion respectively. If the channel is disabled (CHEN = 0), enabling the channel conversion will cause the channel to remain in the conversion state. The channel can be restored only by enabling the channel (CHEN=1) or disabling the HPDF module (HPDFEN=0).

The regular group selects only one of the 8 channels, which is determined by the RCS bit in the HPDF_FLTyCTL0 register. At the same time, only one regular conversion can be executed or pending. If an existing regular conversion request has not been completed, the new regular conversion start request is ignored. The priority of regular conversion is lower than that of inserted group conversion and can be interrupted by inserted group conversion request.

The conversion time of regular group: t = CTCNT[27:0] / $f_{HPDFCLK}$.

#### Conversion start mode

Regular group conversion can only be achieved by software startup. There are two modes of software startup, the specific methods are as follows:

1. General software startup: write 1 to SRCs bit in HPDF_FLTyCTL0 register.
2. Software synchronous start: Set the RCSYN bit in HPDF_FLTyCTL0 register and start the regular conversion of HPDF_FLT0 by general software startup. Then HPDF_FLTy also starts the regular conversion synchronously.

#### Conversion mode

Regular group transformation supports continuous mode and fast mode.

**Continuous mode**

Set the RCCM bit to 1 in HPDF_FLTyCTL0 register to enable continuous mode. In continuous mode, after the software starts the regular group conversion, the conversion regular group channel conversion is repeated. When the RCCM bit is cleared, the regular conversion in continuous mode stops immediately.

**Fast mode**

Enable fast mode by setting FAST bit to 1 in HPDF_FLTyCTL0. In fast mode, it can improve the data rate in continuous mode. Because in continuous mode, if the data is continuously converted from one channel, there is no need to fill the filter with new data, because the data in the filter is valid data sampled from the previous continuous mode. The increase in data rate is determined by the order of the selected filter.

After the continuous conversion is started, the time for the first conversion of the fast mode to the non open fast mode is the same, and then the subsequent conversion will be completed at a shorter time interval.

### 45.3.7. Inserted group conversion

The conversion channel of the inserted group must select at least one of the 8 channels. You can select which channel to convert into the inserted group by the ICGSEL[7:0] bit field in the HPDF_FLTyICGS register. ICGSEL[y]=1 means channel x is the inserted group channel.

The priority of the inserted group is higher than that of the regular group. The ongoing regular group conversion will be interrupted by the inserted group conversion request. Wait for the inserted group to complete the conversion and restart the interrupted regular conversion. At the same time, only one inserted conversion is in execution or pending state. If an existing inserted conversion request has not been completed, the new inserted conversion start request will be ignored.

The conversion time of the inserted group t=CTCNT[27:0] * $f_{HPDFCLK}$.

#### Conversion start mode

The conversion of the inserted group can be achieved through channel software startup and trigger startup.

1. General software startup: write 1 to the SICC bit in HPDF_FLTyCTL0 register.

2. Software synchronous startup: Set the ICSYN bit in HPDF_FLTyCTL0 register to start synchronously. When using general software to start the inserted group conversion of HPDF_FLT0, the channel 1 that enables the synchronous start function also starts the inserted conversion.
3. Trigger startup: When the ICTSSEL[4:0] bit field in the HPDF_FLTyCTL0 register is written with a value other than 0, it indicates that trigger start is enabled and trigger signal source is selected at the same time. The effective edge of the trigger is determined by the ICTEEN[1:0] bit field.

The trigger signals of the inserted group are shown in the following table:

**Table 45-5. Trigger signal of inserted group**

| Trigger signal | Signal source |
|---|---|
| HPDF_ITRG0 | TIMER0_TRGO0 |
| HPDF_ITRG1 | TIMER0_TRGO1 |
| HPDF_ITRG2 | TIMER7_TRGO0 |
| HPDF_ITRG3 | TIMER7_TRGO1 |
| HPDF_ITRG4 | TIMER2_TRGO0 |
| HPDF_ITRG5 | TIMER3_TRGO0 |
| HPDF_ITRG6 | TIMER15_CH1 |
| HPDF_ITRG7 | TIMER5_TRGO0 |
| HPDF_ITRG8 | TIMER6_TRGO0 |
| HPDF_ITRG[9~10] | Reserved |
| HPDF_ITRG11 | TIMER22_TRGO0 |
| HPDF_ITRG12 | TIMER23_TRGO0 |
| HPDF_ITRG[13~23] | Reserved |
| HPDF_ITRG24 | EXTI11 |
| HPDF_ITRG25 | EXTI15 |
| HPDF_ITRG26 | - |
| HPDF_ITRG27 | - |
| HPDF_ITRG28 | - |
| HPDF_ITRG[29~30] | Reserved |
| HPDF_ITRG31 | HPDF_ITRG |

### Scan conversion mode

By setting the SCMOD bit in HPDF_FLTyCTL0 register, the scan conversion mode for inserted group conversion can be enabled. In the scan mode, when the inserted group conversion is triggered, all channels in the inserted group will be converted sequentially starting from the lowest channel.

If the scan mode is disabled, each time the inserted group conversion is triggered, only one channel in the inserted group will be converted, and the next trigger will select another channel. And writing to the ICGSEL[7:0] bit field will use the lowest channel as the selected conversion channel.

### Conversion request priority

The conversion of the inserted group has a higher priority than the regular group. The regular conversion that is already in progress will be immediately interrupted by the inserted conversion request. When the inserted conversion sequence ends, if RCCM remains at 1, the continuous regular conversion will start again.The value of the RCHPDT bit indicates that the interrupted regular conversion is delayed.

If an inserted conversion is pending or already in progress, you cannot start other inserted conversions: as long as ICPF=1, any request to launch an inserted conversion (software or trigger start) will be ignored. The regular conversion is the same.

When the inserted conversion is in progress (ICPF=1), write 1 to the SRCS bit of HPDF_FLTyCTL0 to request regular conversion. When the inserted sequence is completed, the priority indicates the next step to perform regular conversion, and the delayed start is indicated by the RCHPDT bit.

## 45.3.8. Digital filter

The digital filter of the HPDF module is of $Sinc^X$ type. The input data stream is filtered by $Sinc^X$, thereby reducing the output data rate and increasing the output data resolution. Configure the order and oversampling rate (decimation filtering) of the $Sinc^X$ filter by the SFO[2:0] and SFOR[9:0] bits in the HPDF_FLTySFCFG register. The user can configure the order and oversampling rate of the $Sinc^X$ filter according to the desired resolution. The relationship between the maximum output resolution of $Sinc^X$ filtering and oversampling filtering is as follows:

**Table 45-6. The relationship between the maximum output resolution and oversampling filtering of SincX filtering**

| SFOR | Sinc | $Sinc^2$ | FastSinc | $Sinc^3$ | $Sinc^4$ | $Sinc^5$ |
|---|---|---|---|---|---|---|
| x | ±x | $\pm x^2$ | $\pm 2x^2$ | $\pm x^3$ | $\pm x^4$ | $\pm x^5$ |
| 4 | ±4 | ±16 | ±32 | ±64 | ±256 | ±1024 |
| 8 | ±8 | ±64 | ±64 | ±512 | ±4096 | ±32768 |
| 32 | ±32 | ±1024 | ±2048 | ±32768 | ±1048576 | ±33554432 |
| 64 | ±64 | ±4096 | ±8192 | ±262144 | ±16777216 | ±1073741824 |
| 128 | ±128 | ±16384 | ±32768 | ±2097152 | ±268435456 | - |
| 256 | ±256 | ±65536 | ±131072 | ±16777216 | Under full-scale input conditions, the result will overflow | |
| 1024 | ±1024 | ±1048576 | ±2097152 | ±1073741824 | | |

**Note:** The maximum output resolution in this table comes from the peak data value of the filter output.

## 45.3.9. Integrator

The integrator performs further oversampling rate (decimation rate) and resolution

improvement on the data from the digital filter. The integrator performs a simple summation operation on a given number of data samples from the filter. The output data of the integrator comes from the sum of the output samples of the filter, and the number of output samples is determined by the oversampling rate of the integration. The oversampling rate (decimation filtering) of the integrator can be configured by IOR[7:0] in HPDF_FLTySFCFG register. The relationship between the maximum output resolution, oversampling rate, and Sinc filter order of the integrator is as follows:

**Table 45-7. Relationship between the maximum output resolution and IOR, SFOR, SFO of the integrator**

| Filter type | Integrator maximum output resolution |
|---|---|
| Sinc | ±(SFOR×IOR) |
| $Sinc^2$ | ±($SFOR^2$×IOR) |
| FastSinc | ±(2$SFOR^2$×IOR) |
| $Sinc^3$ | ±($SFOR^3$ ×IOR) |
| $Sinc^4$ | ±($SFOR^4$×IOR) |
| $Sinc^5$ | ±($SFOR^5$×IOR) |

## 45.3.10. Threshold monitor

The threshold monitor of the HPDF module is used to monitor the serial input data of the channel or the final output data after the channel conversion. When the data reaches the threshold set by the threshold monitor (maximum or minimum threshold), an interrupt or break event will be generated. The maximum threshold is determined by the HTVAL[23:0] bits in HPDF_FLTyTMHT register, and the minimum threshold is determined by the LTVAL[23:0] bits in HPDF_FLTyTMLT register.

The HPDF module has four threshold monitor. By configuring the TMCHEN[1:0] bit field in HPDF_FLTyCTL1 register, it determines whether the analog threshold monitor x monitors the input channel. For example, TMCHEN[1]=1 in HPDF_FLT0CTL1 register means threshold monitor 0 monitors channel 1.

### Threshold monitor working mode

The working mode of the threshold monitor is divided into standard mode and fast mode. Fast mode is to configure the serial input data of the threshold monitor monitoring channel and compare it with the threshold. Standard mode is to configure the final data output after the threshold monitor monitor channel conversion (stored in the inserted group data register HPDF_FLTyIDATA or the regular group data register HPDF_FLTyRDATA). The fast mode of the threshold monitor can be enabled by the TMFM bit in HPDF_FLTyCTL0. The characteristics in both cases are as follows:

**Table 45-8. Features of threshold monitor working mode**

| Mode | Enable Bit | Channel Data | Analog Input | Input Data | Detailed Description |
|---|---|---|---|---|---|

| | | Source | Data Source | Resolution | |
|---|---|---|---|---|---|
| Standard mode | TMFM=0 | Serial data stream, Parallel data | HPDF final data output | 24bit | The threshold monitor monitors the final data output after the channel conversion. Slow response time, not suitable for overcurrent/overvoltage detection |
| Fast mode | TMFM=1 | Serial data stream | Serial data stream | 16bit | The input data is provided in continuous mode, and the threshold monitor directly monitors the serial input data, regardless of rules or injection conversion. Fast response time, suitable for overcurrent/overvoltage detection. |

In fast mode, the threshold monitor uses only the upper 16 bits of the threshold (maximum threshold HTVAL[23:0] or minimum threshold LTVAL[23:0]) to compare with the serial input data of the channel, that is, only the upper 16 bits of HTVAL[23:0] and LTVAL[23:0] define the threshold, because the resolution of the threshold monitor filter is 16 bits.

In non-fast mode of threshold monitor, the final data of right shift and offset calibration will be compared with HTVAL[23:0] and LTVAL[23:0].

### Threshold monitor fast mode

In fast mode, the filter of the threshold monitor will be used, and the oversampling rate (decimation rate) and order of the threshold monitor filter can be set in HPDF_CHxCFG1 register.

The configuration of the threshold monitor is flexible. An threshold monitor can be configured to monitor multiple channels by the TMCHEN[1:0] bit field in HPDF_FLTyCTL1 register. In this case, when multiple channels send out requests, the threshold monitor will preferentially process requests with small channel numbers, and then process requests with large channel numbers. Each threshold monitor has a status register HPDF_FLTyTMSTAT. When the monitored channel exceeds the threshold, the corresponding flag in the HTF[7:0] or LTF[7:0] bit field will be set. If HTF[0]=2b’01, it means that channel 0 occured an event that exceeds the upper threshold.

After each channel sends a comparison request, it will be executed within 8 HPDF clock cycles. Therefore, the bandwidth of each channel is limited to 8 HPDF clock cycles (if TMCHEN[7:0]=255). Since the maximum sampling frequency of the input channel is

$f_{HPDFCLK}$/4, at this input clock speed, the threshold monitor filter cannot be bypassed (TMFOR=0). Therefore, the user must correctly configure the threshold monitor filter parameters and the number of channels monitored based on the input sampling clock speed and $f_{HPDFCLK}$.

In fast mode, reading the TMDATA[15:0] bit field in HPDF_CHxTMFDT register to get the threshold monitor filter data for the given channel x. The number of serial samples required for a result of the threshold monitor filter output (at the serial input clock frequency $f_{CKIN}$) is as follows:

1. First conversion:

   FastSinc filter: Number of samples is equal to (TMFOR[4:0]×4+ 2+1).

   $Sinc^X$ filter (x=1..5): Number of samples is equal to (((TMFOR[4:0]+1)×TMSFO[1:0])+ TMSFO[1:0]+1).

2. Subsequent conversions other than the first conversion:

   FastSinc and $Sinc^X$ filter (x=1..5): Number of samples is equal to (TMFOR[4:0]+1)×(IOR[7:0]+1).

#### Threshold monitor flag

The global state of the threshold monitor is the TMEOF flag in HPDF_FLTySTAT register. When TMEOF=1, it indicates that at least one threshold monitor event has occurred, that is, an event that exceeds the (upper/lower limit) threshold is generated. If the threshold monitor event interrupt TMIE=1 in HPDF_FLTyCTL1 register is enabled, an threshold monitor interrupt can be generated. When all HTF[7:0] and LTF[7:0] are cleared, the TMEOF bit is cleared.

The HPDF_FLTyTMSTAT register defines the error event flag of the channel exceeding the threshold. The HTF[1:0] bit field indicates whether the maxmum threshold HTVAL[23:0] has been exceeded on the channel x. The LTF[1:0] bit field indicates whether the minimum threshold LTVAL[23:0] value has been exceeded on channel x. Clear the threshold event flag by writing "1" to the corresponding HTFC[7:0] or LTFC[7:0] bit in the HPDF_FLTyTMFC register.

As is shown in ***Table 45-2. HPDF internal signal***, there are 4 break output signals in the HPDF module. The break output signals are assigned to threshold monitor threshold event by setting the HTBSD[3:0] and LTBSD[3:0] bit fields in the HPDF_FLTyTMHT register and the HPDF_FLTyTMLT register.

### 45.3.11. Malfunction monitor

The purpose of the malfunction monitor is to be able to send a signal with an extremely fast response time when the analog signal reaches a saturation value and remains in this state for a given time. This feature can be used to detect short-circuit or open-circuit faults (e.g. overcurrent/overvoltage). The broken output signals can be assigned to the malfunction monitor event, which can be configured by the MMBSD[3:0] bit field in the HPDF_CHxCFG1

register. The broken output signal is the same as the threshold monitor.

The input data of the malfunction monitor comes from the serial input data of the channel.When the channel input data source is parallel data, the malfunction monitor function is prohibited.There is an up counter on each input channel to record how many consecutive 0 or 1 on the output of the serial data receiver. When the counter reaches the threshold value of the malfunction monitor (MMCT[7:0] bits in the HPDF_CHxCFG1 register), a malfunction event occurs. If a 0-1 or 1-0 change is encountered when monitoring the data stream, the value of the counter will be automatically cleared and counted again.

The user can enable the malfunction monitor function by setting the MMEN bit in HPDF_CHxCTL register. When a malfunction event occurs on the channel, the corresponding malfunction monitor flag MMF[1:0] is set. The corresponding flag can be cleared by MMFC[1:0] in HPDF_FLTyINTC. If channel x is disabled (CHEN=0), the hardware will also clear the malfunction monitor flag.

### 45.3.12. Extremes monitor

The extremes monitor is used to sample the minimum and maximum values (peak to peak) of the final output data word. An extremes monitor can be configured to sample the extreme values of multiple channels through the EMCS[7:0] bit field in HPDF_FLTyCTL1 register.

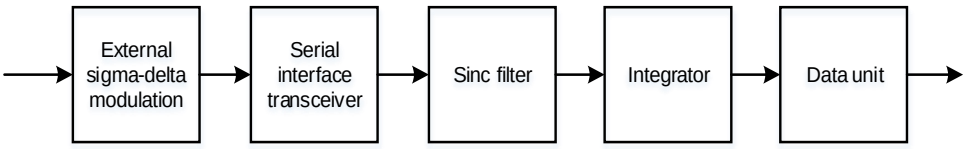
If the sampling final output data word is higher than the value in the maximum value register of the extremes monitor (MAXVAL[23:0] bits in the HPDF_FLTyEMMAX register), the value of this register is updated to the current final output data. If the sampling final output data word is smaller than the value in the minimum value register of the extremes monitor (MINVAL[23:0] bits in HPDF_FLTyEMMIN register), the value of this register is updated to the current final output data. The values of the MAXDC bit and the MINDC bit indicate which channel the maximum/minimum value comes from.

When reading the HPDF_FLTyEMMAX or HPDF_FLTyEMMIN register, the maximum or minimum value is updated with the reset value.

### 45.3.13. Data unit

The data unit is the last part of data processing in the entire HPDF module, and the flow of data processing by the HPDF module is shown in the following figure.

**Figure 45-7. HPDF module external input data processing flow**




The output data rate depends on the serial data stream rate, filter and integrator settings. The maximum output data rate is shown in the table below.

**Table 45-9. Maximum output rate**

| Input source | Conversion mode | Filter type | Maximum output data rate (samples/second) |
|---|---|---|---|
| Serial input | Non-fast mode (FAST=0) | $Sinc^X$ | $\frac{f_{CKIN}}{SFOR\times(IOR-1+SFO)+(SFO+1)}$ |
| | Non-fast mode (FAST=0) | FastSinc | $\frac{f_{CKIN}}{SFOR\times(IOR-1+4)+(2+1)}$ |
| | Fast mode (FAST=1) | FastSinc and $Sinc^X$ | $\frac{f_{CKIN}}{SFOR\times IOR}$ |
| Parallel input | Non-fast mode (FAST=0) | $Sinc^X$ | $\frac{f_{DATA}}{SFOR\times(IOR-1+SFO)+(SFO+1)}$ |
| | Non-fast mode (FAST=0) | FastSinc | $\frac{f_{DATA}}{SFOR\times(IOR-1+4)+(2+1)}$ |
| | Fast mode (FAST=1) | FastSinc and $Sinc^X$ | $\frac{f_{DATA}}{SFOR\times IOR}$ |

**Note:** $f_{DATA}$ is the parallel data rate of the CPU/DMA input. When the filter is bypassed, $f_{DATA} \leqslant f_{HPDFCLK}$ must be satisfied.

### Signed data format

Signed data in HPDF module: parallel data register, regular and inserted group data register, threshold monitor vlaue, extreme monitor value, and offset calibration are all signed formats. The most significant bit of the output data indicates the sign of the value, and the data is in two's complement format.

Since all operations in digital processing are performed on 32-bit signed registers, the following conditions must be met in order for the result not to overflow:

1. When using $Sinc^X$ filter (x=1..5): $(SFOR^{SFO})\times IOR \leq 2^{31}$.
2. When using FastSinc filter: $2\times(SFOR^2)\times IOR \leq 2^{31}$.

### Data right bit shift

Since the final data width is 24 bits and the data from the processing path can be up to 32 bits, the right shift of the final data is performed in this module. For each selected input channel, the number of bits shifted to the right can be configured in the DTRS[4:0] bit field in the HPDF_CHxCFG0 register. The result is round to the nearest value by abandoning the lowest bit.

### Data offset calibration

In the HPDF module, each channel has a data offset calibration value, which is stored in the CALOFF[23:0] bit field in HPDF_CHxCFG0 register. When offset calibration is performed, the offset calibration value is subtracted from the output data of the channel to obtain the final data output by the HPDF module.

Data offset calibration occurs after the right shift of the data.

### 45.3.14. HPDF interrupt

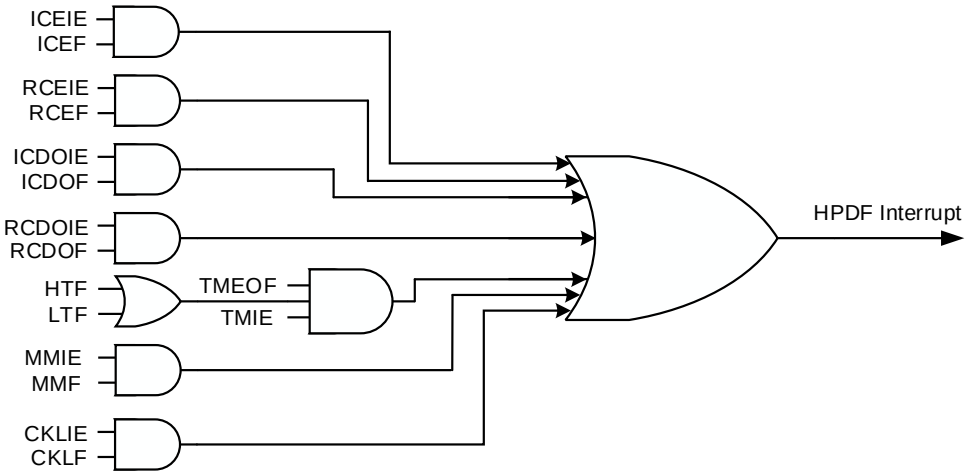
HPDF interrupt events can be divided into channel conversion interrupt events, threshold monitor interrupt events, malfunction monitor interrupt events, and channel clock loss interrupt events. The specific interrupt event description is as ***Table 45-10. HPDF interrupt event***.

**Table 45-10. HPDF interrupt event**

| Interrupt event | description | Clear | Enable interrupt |
|---|---|---|---|
| ICEF | end of inserted conversion | Read HPDF_FLTyIDATA register | ICEIE |
| RCEF | end of regular conversion | Read HPDF_FLTyRDATA register | RCEIE |
| ICDOF | inserted conversion data overflow | Write 1 to the ICDOFC bit | ICDOIE |
| RCDOF | regular conversion data overflow | Write 1 to RCDOFC bit | RCDOIE |
| TMEOF<br>HTF[7:0]<br>LTF[7:0] | threshold monitor events | Write 1 to HTFC[7:0] bit field<br>Write 1 to LTFC[7:0] bit field | TMIE |
| MMF | malfunction event | Write 1 to MMFC[7:0] bits | MMIE |
| CKLF | Channel clock loss event | Write 1 to CKLFC[7:0] bits | CKLIE |

HPDF interrupt logic is as ***Figure 45-8. HPDF interrupt logic diagram*** shown.

**Figure 45-8. HPDF interrupt logic diagram**



## 45.4. Register definition

HPDF base address: 0x4001 7000

### 45.4.1. HPDF channel x registers (x=0, 7)

#### Channel x control register (HPDF_CHxCTL)

Address offset: 0x00 + 0x20 * x, (x = 0, 7)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HPDFEN | CKOUTSEL | CKOUTDM | Reserved | | | | | CKOUTDIV[7:0] | | | | | | | |
| rw | rw | rw | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DPM[1:0] | | CMSD[1:0] | | Reserved | | | CHPINSEL | CHEN | CKLEN | MMEN | Reserved | SPICKSS[1:0] | | SITYP[1:0] | |
| rw | | rw | | | | | rw | rw | rw | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | HPDFEN | Global enable for HPDF interface<br>0: HPDF disabled<br>1: HPDF enabled<br>If HPDFEN=0, the HPDF_FLTySTAT register and HPDF_FLTyTMSTAT register is set to reset state.<br>This bit is only available in HPDF_CH0CTL. |
| 30 | CKOUTSEL | Serial clock output source selection<br>0: Serial clock output source is from CK_HPDF clock<br>1: Serial clock output source is from CK_HPDFAUDIO clock<br>This bit can be configured only when HPDFEN=0.<br>This bit is only available in HPDF_CH0CTL. |
| 29 | CKOUTDM | Serial clock output duty mode<br>0: Serial clock output duty mode disable<br>1: Serial clock output duty mode enable, the duty is 1:1<br>This bit can be configured only when HPDFEN=0.<br>This bit is only available in HPDF_CH0CTL. |
| 28:24 | Reserved | Must be kept at reset value. |
| 23:16 | CKOUTDIV[7:0] | Serial clock output divider<br>0: Output clock generation is disabled (CKOUT signal is set to low state)<br>1~255: The value of division for the serial clock output is CKOUTDIV+1. |

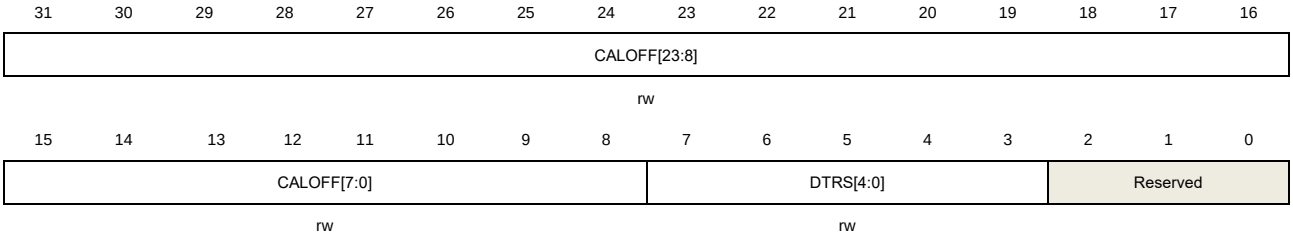
| | | |
|---|---|---|
| | | CKOUTDIV also defines the threshold for a clock loss detection.<br>This value can only be modified when HPDFEN=0.During a HPDF clock cycle after HPDFEN=0, the CKOUT is set to low state.<br>This bit is only available in HPDF_CH0CTL. |
| 15:14 | DPM[1:0] | Data packing mode for HPDF_CHxPDI register<br>00: Standard mode<br>01: Interleaved mode<br>10: Dual mode<br>11: Reserved<br>For a detailed introduction of data encapsulation mode, please refer to ***Parallel data packed mode.***<br>These bits can be configured only when CHEN=0. |
| 13:12 | CMSD[1:0] | Channel x multiplexer select input data source<br>00: Input data source for channel x is taken from serial inputs<br>01: Input data source for channel x is taken from the internal analog-to-digital converter ADC output register update<br>10: Input data source for channel x is taken from internal HPDF_CHxPDI register<br>11: Reserved<br>The HPDF_CHxPDI register is write protected when these bits are reset.<br>These bits can be configured only when CHEN=0. |
| 11:9 | Reserved | Must be kept at reset value. |
| 8 | CHPINSEL | Channel inputs pins selection<br>0: Channel inputs select pins of the current channel x<br>1: Channel inputs select pins of the next channel.<br>This bit can be configured only when CHEN=0. |
| 7 | CHEN | Channel x enable<br>0: Channel x disabled<br>1: Channel x enabled<br>If channel x is enabled, then serial data will be received based on the given channel settings. |
| 6 | CKLEN | Clock loss detector enable<br>0: Clock loss detector disabled<br>1: Clock loss detector enabled |
| 5 | MMEN | Malfunction monitor enable<br>0: malfunction monitor is no effect<br>1: malfunction monitor is effect |
| 4 | Reserved | Must be kept at reset value. |
| 3:2 | SPICKSS[1:0] | SPI clock source select<br>00: External CKINx input is selected for SPI clock source, sampling point is determined by SITYP[1:0] |

| | | |
|---|---|---|
| | | 01: Internal CKOUT output is selected for SPI clock source, sampling point is determined by SITYP[1:0]<br>10: Internal CKOUT is selected for SPI clock source, sampling point on each second CKOUT falling edge.<br>11: Internal CKOUT output is selected for SPI clock source, sampling point on each second CKOUT rising edge.<br>These bits can be configured only when CHEN=0. |
| 1:0 | SITYP[1:0] | Serial interface type<br>00: SPI interface, sample data on rising edge<br>01: SPI interface, sample data on falling edge<br>10: Manchester coded input: rising edge = logic 0, falling edge = logic 1<br>11: Manchester coded input: rising edge = logic 1, falling edge = logic 0<br>These bits can only be configured when CHEN=0. |

### Channel x configuration register 0 (HPDF_CHxCFG0)

Address offset: 0x04 + 0x20 * x, (x = 0, 7)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

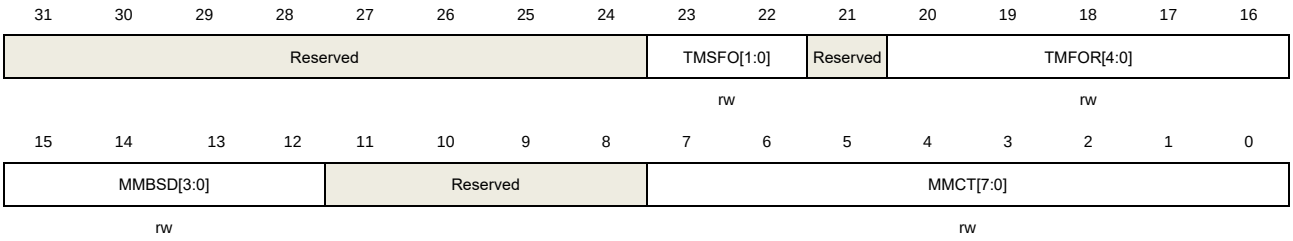



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | CALOFF[23:0] | 24-bit calibration offset<br>Calibration offset must be performed for each conversion result of the channel.<br>These bits can be set by software. |
| 7:3 | DTRS[4:0] | Data right bit-shift<br>0-31: The number of bits that determine the right shift of data<br>Bit-shift is performed before offset correction. The data shift rounds the result to the nearest integer value and the sign is preserved.<br>These bits can be configured only when CHEN=0 (in HPDF_CHxCTL register). |
| 2:0 | Reserved | Must be kept at reset value. |

### Channel x configuration register 1 (HPDF_CHxCFG1)

Address offset: 0x08 + 0x20 * x, (x = 0, 7)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

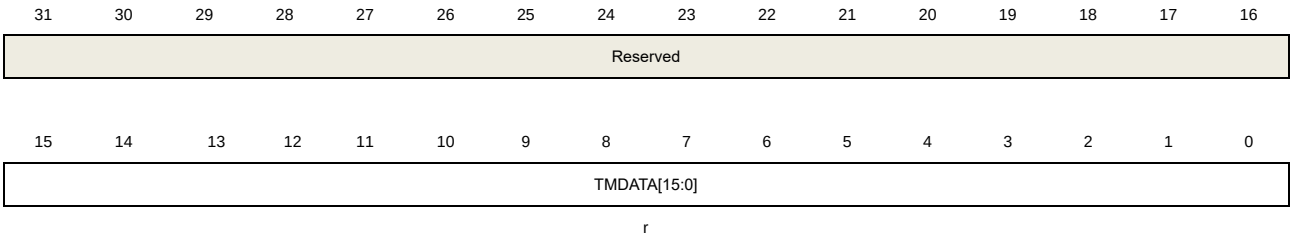



| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:22 | TMSFO[1:0] | Threshold monitor Sinc filter order selection<br>00: FastSinc filter<br>01: Sinc$^1$ filter<br>10: Sinc$^2$ filter<br>11: Sinc$^3$ filter<br>These bits can be configured only when CHEN=0 (in HPDF_CHxCTL register). |
| 21 | Reserved | Must be kept at reset value |
| 20:16 | TMFOR[4:0] | Threshold monitor filter oversampling rate (decimation rate)<br>0 - 31: The filter decimation rate equal to TMFOR[4:0]+ 1<br>If TMFOR=0, the filter is bypassed.<br>These bits can be configured only when CHEN=0 (in HPDF_CHxCTL register). |
| 15:12 | MMBSD[3:0] | Malfunction monitor break signal distribution<br>MMBSD[i] = 0: Break i signal not is distributed to malfunction monitor on channel x<br>MMBSD[i] = 1: Break i signal is distributed to malfunction monitor on channel x |
| 11:8 | Reserved | Must be kept at reset value. |
| 7:0 | MMCT[7:0] | Malfunction monitor counter threshold<br>These bits be used determine the count value of malfunction monitor counter threshold. The count value is written by software.<br>If the count value is reached, then an event of malfunction monitor occurs on a given channel. |

### Channel x threshold monitor filter data register (HPDF_CHxTMFDT)

Address offset: 0x0C + 0x20 * x, (x = 0, 7)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | TMDATA[15:0] | Threshold monitor data<br>The data is come from the threshold monitor filter and continuously converted (no trigger) for this channel. |

### Channel x parallel data input register (HPDF_CHxPDI)

Address offset: 0x10 + 0x20 * x, (x = 0, 7)
Reset value: 0x0000 0000

This register has to be accessed by half-word (16-bit) and word (32-bit).

This register contains 16-bit input data to be processed by HPDF filter module.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATAIN1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DATAIN0[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | DATAIN1[15:0] | Data input for channel x or channel x+1<br>Data can be written by CPU/DMA.<br>If DPM[1:0]=0 (standard mode), DATAIN1[15:0] is write protected.<br>If DPM[1:0]=1 (interleaved mode), second channel x data sample is stored into DATAIN1[15:0]. First channel x data sample is stored into DATAIN0[15:0]. Both samples are read sequentially by HPDF_FLTy filter.<br>If DPM[1:0]=2 (dual mode):<br>For channel 0: sample in DATAIN1[15:0] is automatically copied into DATAIN0[15:0] of channel 1.<br>For channel 1: DATAIN1[15:0] is write protected.<br>The more details refer to ***Parallel data packed mode***<br>DATAIN1[15:0] is a signed format data. |
| 15:0 | DATAIN0[15:0] | Data input for channel x<br>Data can be written by CPU/DMA.<br>If DPM[1:0]=0 (standard mode), channel x data sample is stored into DATAIN0[15:0].<br>If DPM[1:0]=1 (interleaved mode), first channel x data sample is stored into DATAIN0[15:0]. Second channel x data sample is stored into DATAIN1[15:0]. Both samples are read sequentially by HPDF_FLTy filter.<br>If DPM[1:0]=2 (dual mode):<br>For channel 0: Channel x data sample is stored into DATAIN0[15:0].<br>For channel 1: DATAIN0[15:0] is write protected. |

The more details refer to ***Parallel data packed mode***

DATAIN0[15:0] is a signed format data.

### Channel x pulse skip register (HPDF_CHxPS)

Address offset: 0x14 + 0x20 * x, (x = 0, 7)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | PLSK[5:0] | | | | | |
| | | | | | | | | | | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:6 | Reserved | Must be kept at reset value. |
| 5:0 | PLSK[5:0] | Pulses to skip for input data skipping function<br>0-63: Defines the number of serial input samples that will be skipped.<br>Skipping function is take effect immediately after writing to this field. Read PLSK[5:0] to return the remaining value of the pulses which will be skipped.<br>The value of PLSK[5:0] can be updated even PLSK[5:0] is not zero. |

## 45.4.2. HPDF filter y registers (y=0, 3)

### Filter y control register 0 (HPDF_FLTyCTL0)

Address offset: 0x100 + 0x80 * y, (y = 0, 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | TMFM | FAST | Reserved | | RCS[2:0] | | | Reserved | | RCDMAEN | Reserved | RCSYN | RCCM | SRCS | Reserved |
| | rw | rw | | | rw | | | | | rw | | rw | rw | rt_w | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | ICTEEN[1:0] | | ICTSSEL[4:0] | | | | | Reserved | | ICDMAEN | SCMOD | ICSYN | Reserved | SICC | FLTEN |
| | rw | | rw | | | | | | | rw | rw | rw | | rt_w | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | TMFM | Threshold monitor fast mode |

| | | |
|---|---|---|
| | | 0: Threshold monitor watch on the final data after performing offset correction and right shift<br>1: Threshold monitor watch on serial input data stream |
| 29 | FAST | Fast conversion mode for regular conversions<br>0: Fast conversion mode disabled<br>1: Fast conversion mode enabled<br>If fast mode is enabled, the normal conversion in continuous mode (except for the first conversion) is performed faster than the conversion in standard mode. This bit has no effect on conversions which are not continuous.<br>This bit can be configured only when FLTEN=0. |
| 28:27 | Reserved | Must be kept at reset value |
| 26:24 | RCH[2:0] | Regular conversion channel selection<br>0: Channel 0 is selected as the regular conversion channel<br>1: Channel 1 is selected as the regular conversion channel<br>…<br>7: Channel 7 is selected as the regular conversion channel<br>When RCPF=1, writing this bit takes effect when the next regular conversion begins. |
| 23:22 | Reserved | Must be kept at reset value |
| 21 | RCDMAEN | DMA channel enabled to read data for the regular conversion<br>0: Disable the DMA channel to read regular data<br>1: Enable the DMA channel to read regular data<br>This bit can be configured only when FLTEN=0. |
| 20 | Reserved | Must be kept at reset value. |
| 19 | RCSYN | Regular conversion synchronously with HPDF_FLT0<br>0: Do not launch a regular conversion synchronously with HPDF_FLT0<br>1: Launch a regular conversion synchronously in HPDF_FLTy when a regular conversion is launched in HPDF_FLT0<br>If RCSYN=1 in HPDF_FLT0CTL0 register, the regular conversion channel will be Launched synchronously which selected in HPDF_FLTyCTL0.<br>This bit can be configured only when FLTEN=0. |
| 18 | RCCM | Regular conversions continuous mode<br>0: The regular channel is converted just once for each conversion request<br>1: The regular channel is converted repeatedly after each conversion request<br>Writing "0" to this bit will immediately stop continuous mode during a continuous regular conversion. |
| 17 | SRCS | Start regular channel conversion by software<br>0: No effect<br>1: Make a request to start regular channel conversion<br>If RCPF=1, invalid write to SRCS, and if RCSYN=1, write ‘1’ to SRCS, launch a regular conversion synchronously. |

| | | |
|---|---|---|
| | | This bit is always read as '0'. |
| 16:15 | Reserved | Must be kept at reset value. |
| 14:13 | ICTEEN[1:0] | Inserted conversions trigger edge enable<br>00: Disable trigger detection<br>01: Each rising edge on the trigger signal makes a request to start an inserted conversion<br>10: Each falling edge on the trigger signal makes a request to start an inserted conversion<br>11: The edge (rising edges and falling edges) on the trigger signal make requests to start inserted conversions<br>This bit can be configured only when FLTEN=0. |
| 12:8 | ICTSSEL[4:0] | Inserted conversions trigger signal selection<br>0x00~0x1F: The value indicates that different trigger signals are selected to start the conversion<br>The maximum delay from the generation of trigger signal to the start of synchronous trigger is 1 $f_{HPDFCLK}$ clock cycle, and the delay of asynchronous trigger is 2-3 $f_{HPDFCLK}$ clock cycles.<br>This bit can be configured only when FLTEN=0. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | ICDMAEN | DMA channel enabled to read data for the inserted channel group<br>0: Disable DMA channel to read inserted conversions data<br>1: Enable DMA channel to read inserted conversions data<br>This bit can be configured only when FLTEN=0. |
| 4 | SCMOD | Scan conversion mode of inserted conversions<br>0: One channel conversion is performed from the inserted channel group and next the channel is selected from this group.<br>1: The series of conversions for the inserted group channels is executed, starting over with the lowest selected channel.<br>If SCMOD=0, writing ICGSEL will resets the channel selection to the lowest selected channel.<br>This bit can be configured only when FLTEN=0. |
| 3 | ICSYN | Inserted conversion synchronously<br>0: Do not launch an inserted conversion synchronously with HPDF_FLT0<br>1: Launch an inserted conversion synchronously in HPDF_FLTy when an inserted conversion is launched by trigger SICC.<br>This bit can be configured only when FLTEN=0. |
| 2 | Reserved | Must be kept at reset value |
| 1 | SICC | Start inserted group channel conversion<br>0: No effect.<br>1: Makes a request to convert the channels in the inserted conversion group. |

| | | |
|---|---|---|
| | | If ICPF=1 already, invalid write to SICC. If RCSYN=1, write '1' to SICC, launch an inserted conversion synchronously.<br>This bit is always read as '0'. |
| 0 | FLTEN | HPDF_FLTy enable<br>0: HPDF_FLTy is disabled.<br>1: HPDF_FLTy is enabled.<br>If HPDF_FLTy is enabled, then HPDF_FLTy starts operating according to its setting.<br>If HPDF_FLTy is disabled, all conversions of given HPDF_FLTy are stopped immediately and all HPDF_FLTy functions are stopped. Meanwhile HPDF_FLTySTAT register and HPDF_FLTyTMSTAT register is set to the reset state. |

### Filter y control register 1 (HPDF_FLTyCTL1)

Address offset: 0x104 + 0x80 * y, (y = 0, 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TMCHEN[7:0] | | | | | | | |
| | | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EMCS[7:0] | | | | | | | | Reserved | CKLIE | MMIE | TMIE | RCDOIE | ICDOIE | RCEIE | ICEIE |
| rw | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | TMCHEN[7:0] | Threshold monitor channel enable<br>These bits select the input channel to be guarded continuously by the threshold monitor.<br>TMCHEN[x] = 0: Threshold monitor y is disabled on channel x<br>TMCHEN[x] = 1: Threshold monitor y is enabled on channel x |
| 15:8 | EMCS[7:0] | Extremes monitor channel selection<br>These bits select the input channels to be taken by the extremes monitor.<br>EMCS[x] = 0: Extremes monitor y does not monitor data from channel x<br>EMCS[x] = 1: Extremes monitor y monitor data from channel x |
| 7 | Reserved | Must be kept at reset value |
| 6 | CKLIE | Clock loss interrupt enable<br>0: Detection of channel input clock loss interrupt is disabled<br>1: Detection of channel input clock loss interrupt is enabled<br>This bit is only available in HPDF_FLT0CTL1 register. |
| 5 | MMIE | Malfunction monitor interrupt enable<br>0: malfunction monitor interrupt is disabled |

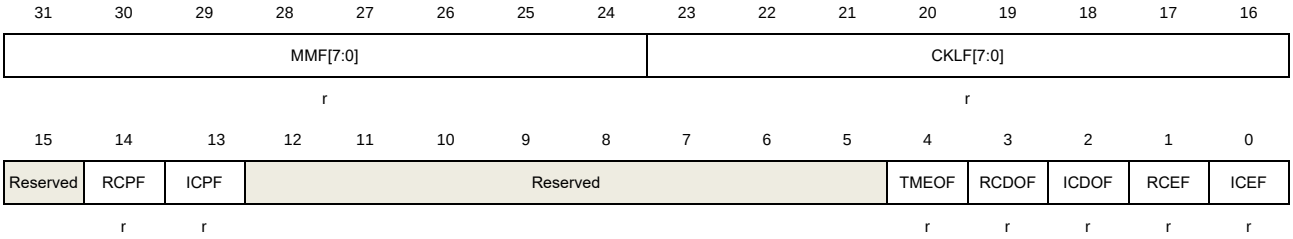
| | | |
|---|---|---|
| | | 1: malfunction monitor interrupt is enabled<br>This bit is only available in HPDF_FLT0CTL1 register. |
| 4 | TMIE | Threshold monitor interrupt enable<br>0: Threshold monitor interrupt is disabled<br>1: Threshold monitor interrupt is enabled |
| 3 | RCDOIE | Regular conversion data overflow interrupt enable<br>0: Regular conversion data overflow interrupt is disabled<br>1: Regular conversion data overflow interrupt is enabled |
| 2 | ICDOIE | Inserted conversion data overflow interrupt enable<br>0: Inserted data overflow interrupt is disabled<br>1: Inserted data overflow interrupt is enabled |
| 1 | RCEIE | Regular conversion end interrupt enable<br>0: Regular conversion end interrupt is disabled<br>1: Regular conversion end interrupt is enabled |
| 0 | ICEIE | Inserted conversion end interrupt enable<br>0: Inserted conversion end interrupt is disabled<br>1: Inserted conversion end interrupt is enabled |

### Filter y status register (HPDF_FLTySTAT)

Address offset: 0x108 + 0x80 * y, (y = 0, 3)
Reset value: 0x0003 0000

This register has to be accessed by word (32-bit).

All the bits of HPDF_FLTySTAT are automatically reset when FLTEN=0.




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | MMF[7:0] | Malfunction monitor flag<br>MMF[x]=0: No malfunction event occurred on channel x<br>MMF[x]=1: Malfunction event occurred on channel x<br>This bit is set by hardware. It can be cleared by software using the corresponding MMFC[x] bit in the HPDF_ FLTyINTC register. MMF[x] is cleared also by hardware when CHEN[x] = 0 (given channel is disabled).<br>This bit is only available in HPDF_FLT0STAT register. |
| 23:16 | CKLF[7:0] | Clock loss flag |

| | | |
|---|---|---|
| | | CKLF[x]=0: Clock signal is not lost on channel x<br>CKLF[x]=1: Clock signal is lost on channel x<br>When CHEN=0 or the serial interface is not synchronized, the state is maintained by the hardware. After the synchronization of serial interface is completed, if the clock of channel x is lost, the corresponding bit in CKLF[7:0] bit field are set by hardware. By setting the CKLFC[7:0] bit field in HPDF_FLTyINTC, the corresponding bit in the CKLF[7:0] bit field can be cleared.<br>This bit is only available in HPDF_FLT0STAT register. |
| 15 | Reserved | Must be kept at reset value |
| 14 | RCPF | Regular conversion in progress flag<br>0: No request of regular conversion has been generated<br>1: The regular conversion is in progress or a request for a regular conversion is pending<br>If RCPF=1, a request to start a regular conversion is ignored. When write 1 to SRCS bit, the RCPF will be setted 1 immediately. |
| 13 | ICPF | Inserted conversion in progress flag<br>0: No request to the inserted group conversion has been generated (neither by software nor by trigger).<br>1: The inserted group conversion is in progress or a request for a inserted conversion is pending.<br>If ICPF=1, a request to start an inserted conversion is ignored. When write 1 to SICC bit, the ICPF will be setted 1 immediately. |
| 12:5 | Reserved | Must be kept at reset value |
| 4 | TMEOF | Threshold monitor event occurred flag<br>0: No Threshold monitor event occurred.<br>1: Threshold monitor event occurred which detected data crosses the threshold<br>This bit is set by hardware.<br>It is cleared by clearing HTF[7:0] and LTF[7:0] in HPDF_FLTyTMSTAT register. |
| 3 | RCDOF | Regular conversion data overflow flag<br>0: No regular conversion data overflow has occurred<br>1: A regular conversion data overflow has occurred<br>If RCDOF=1, it means that a regular conversion finished while RCEF has already been set. RDATA is not affected by overflows.<br>This bit is set by hardware.<br>It can be cleared by setting the RCDOFC in the HPDF_FLTyINTC register. |
| 2 | ICDOF | Inserted conversion data overflow flag<br>0: No inserted conversion data overflow has occurred<br>1: An inserted conversion data overflow has occurred<br>If RCDOF=1, it means that an inserted conversion finished while ICEF has already been set. FLTyIDATA is not affected by overflows<br>This bit is set by hardware. |

| | | |
|---|---|---|
| | | It can be cleared by software setting the ICDOFC bit in the HPDF_FLTyINTC register. |
| 1 | RCEF | Regular conversion end flag<br>0: No regular conversion has completed<br>1: A regular conversion has completed<br>If RCEF=1, it means that the data may be read.<br>This bit is set by hardware. It is cleared when the software or DMA reads HPDF_FLTyRDATA register. |
| 0 | ICEF | Inserted conversion end flag<br>0: No inserted conversion has completed<br>1: An inserted conversion has completed<br>If ICEF=1, it means that its data may be read.<br>This bit is set by hardware. It is cleared when the software or DMA reads HPDF_FLTyIDATA register. |

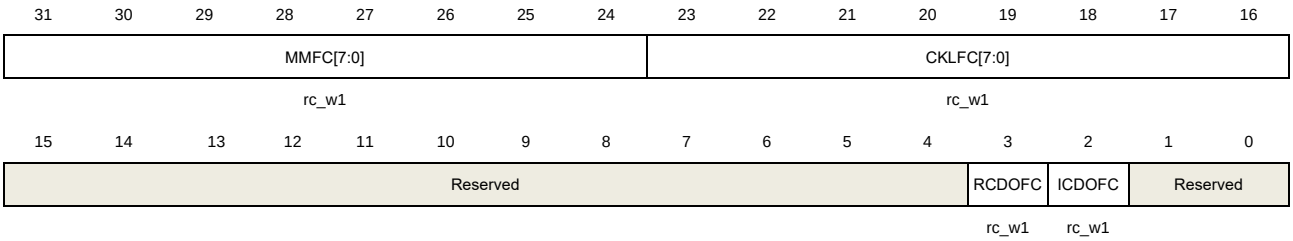
### Filter y interrupt flag clear register (HPDF_FLTyINTC)

Address offset: 0x10C + 0x80 * y, (y = 0, 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

Note: The bits of HPDF_FLTyINTC are always read as '0'.



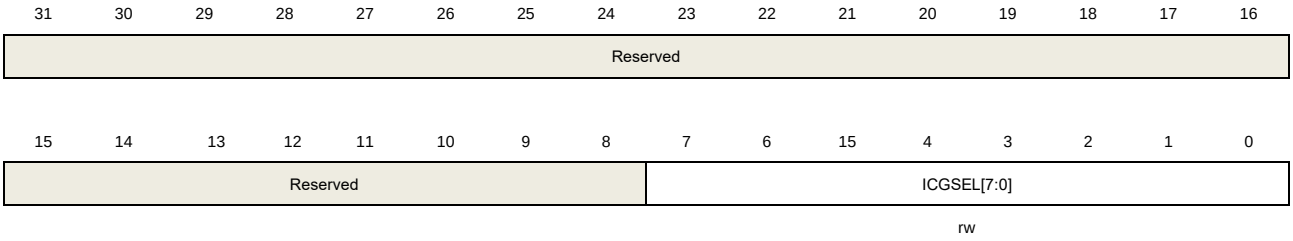

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | MMFC[7:0] | Clear the malfunction monitor flag<br>MMFC[x]=0: No effect<br>MMFC[x]=1: Clear the malfunction monitor flag on channel x<br>This bit is only available in HPDF_FTL0INTC register. |
| 23:16 | CKLFC[7:0] | Clear the clock loss flag<br>CKLFC[x]=0: No effect<br>CKLFC[x]=1: Clear the clock loss flag on channel x<br>When the serial transceiver is not yet synchronized, the clock loss flag is set and cannot be cleared by CKLFC[7:0].<br>This bit is only available in HPDF_FTL0INTC register. |
| 15:4 | Reserved | Must be kept at reset value. |
| 3 | RCDOFC | Clear the regular conversion data overflow flag |

| | | |
|---|---|---|
| | | 0: No effect<br>1: Clear the RCDOF bit in the HPDF_FLTySTAT register |
| 2 | ICDOFC | Clear the inserted conversion data overflow flag<br>0: No effect<br>1: Clear the ICDOF bit in the HPDF_FLTySTAT register |
| 1:0 | Reserved | Must be kept at reset value. |

### Filter y inserted channel group selection register (HPDF_FLTyICGS)

Address offset: 0x110 + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0001

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | ICGSEL[7:0] | Injected channel group selection<br>ICGSEL[x]=0: Channel x is not belongs to the inserted group<br>ICGSEL[x]=1: Channel x belongs to the inserted group<br>If SCMOD=1, each of the selected channels is converted, one after another. The priority conversion with lowest channel number.<br>If SCMOD=0, then only one channel is converted from the selected channels, and the channel selection is moved to the next channel. When SCMOD=0, Writing ICGSEL will reset the channel selection to the lowest selected channel.<br>At least one channel must always be selected for the inserted group. All writes that make ICGSEL[7:0]=0 are ignored. |

### Filter y sinc filter configuration register (HPDF_FLTySFCFG)

Address offset: 0x114 + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| Reserved | IOR[7:0] |
|---|---|
| | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | SFO[2:0] | Sinc filter order.<br>000: FastSinc filter type<br>001: Sinc$^1$ filter type<br>010: Sinc$^2$ filter type<br>011: Sinc$^3$ filter type<br>100: Sinc$^4$ filter type<br>101: Sinc$^5$ filter type<br>110~111: Reserved<br>This bit can only be configured when FLTEN=0 in HPDF_FLTyCTL0 register. |
| 28:26 | Reserved | Must be kept at reset value. |
| 25:16 | SFOR[9:0] | Sinc filter oversampling ratio (decimation rate)<br>0 ~1023: Sinc filter oversampling ratio (decimation rate) SFOR= SFOR[9:0] +1.<br>If SFOR [9:0] = 0 (SFOR=1), the filter will be bypass.<br>This bit can only be configured when FLTEN=0 in HPDF_FLTyCTL0 register. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | IOR[7:0] | Integrator oversampling ratio<br>0~255: Integrator oversampling ratio IOR=IOR[7:0]+1.<br>The output data rate from the integrator will be decreased by this value.<br>If IOR[7:0] = 0 (IOR=1), the integrator will be bypass.<br>This bit can only be configured when FLTEN=0 in HPDF_FLTyCTL0 register. |

### Filter y inserted group conversion data register (HPDF_FLTyIDATA)

Address offset: 0x118 + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

Note: Half-word accesses may be used to read only the MSB of conversion data. DMA can be used to read the data from this register. Reading this register also clears ICEF bit.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDATA[23:8] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IDATA[7:0] | | | | | | | | Reserved | | | | | | ICCH[2:0] | |
| r | | | | | | | | | | | | | | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | IDATA[23:0] | Inserted group conversion data |

| | | |
|---|---|---|
| | | When each a channel in the inserted group is converted, the resulting data is stored in this field.<br>The data is valid when ICEF=1. Reading this register clears the corresponding ICEF (in HPDF_FLTySTAT register). |
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | ICCH[2:0] | Inserted channel most recently converted<br>When each a channel in the inserted group is converted, ICCH is updated to indicate which channel was converted. Therefore, IDATA[23:0] holds the data that corresponds to the channel indicated by ICCH. |

### Filter y regular channel conversion data register (HPDF_FLTyRDATA)

Address offset: 0x11C + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by half-word (16-bit). or word (32-bit).

Note: Half-word accesses may be used to read only the MSB of conversion data. Reading this register also clears RCEF bit.

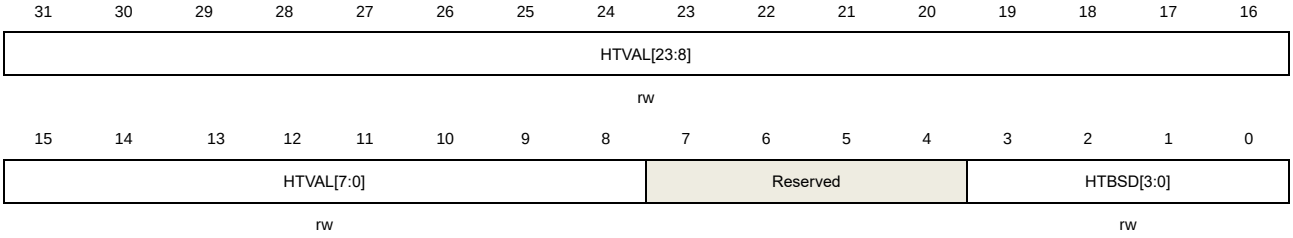



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | RDATA[23:0] | Regular channel conversion data<br>When each regular conversion finishes, its data is stored in these bits. The data is valid when RCEF=1. Reading this register clears the corresponding RCEF (in HPDF_FLTySTAT register). |
| 7:5 | Reserved | Must be kept at reset value. |
| 4 | RCHPDT | Regular channel pending data<br>Regular data in RDATA[23:0] was delayed due to an inserted channel trigger during the conversion |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | RCCH[2:0] | Regular channel most recently converted<br>When each regular conversion finishes, RCCH[2:0] is updated to indicate which channel was converted. Thus RDATA[23:0] holds the data that corresponds to the channel indicated by RCCH[2:0]. |

### Filter y threshold monitor high threshold register (HPDF_FLTyTMHT)

Address offset: 0x120 + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

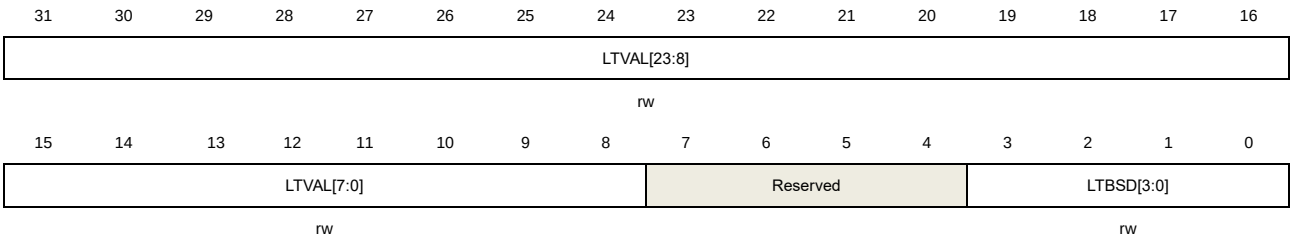



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | HTVAL[23:0] | Threshold monitor high threshold value<br>These bits are written by software to determine the high threshold for the threshold monitor.<br>If TMFM=1, the higher 16 bits determine the 16-bit threshold as compared with the threshold monitor filter output. Bits HTVAL[7:0] are ignored. |
| 7:4 | Reserved | Must be kept at reset value. |
| 3:0 | HTBSD[3:0] | High threshold event break signal distribution<br>HTBSD[i] = 0: Break signal is not distributed to high threshold event<br>HTBSD[i] = 1: Break signal x is distributed to high threshold event |

### Filter y threshold monitor low threshold register (HPDF_FLTyTMLT)

Address offset: 0x124 + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | LTVAL[23:0] | Threshold monitor low threshold value.<br>These bits are written by software to determine the low threshold for the threshold monitor.<br>If TMFM=1, the higher 16 bits determine the 16-bit threshold as compared with the threshold monitor filter output. Bits LTVAL[7:0] are ignored. |
| 7:4 | Reserved | Must be kept at reset value. |

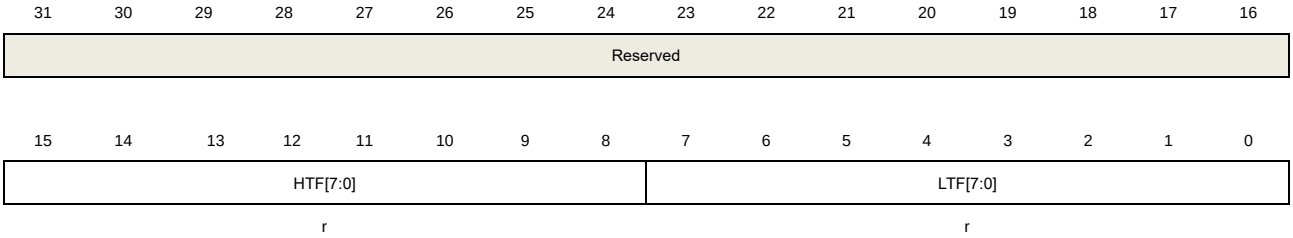
| | | |
|---|---|---|
| 3:0 | LTBSD[3:0] | Low threshold event break signal distribution.<br>LTBSD[i] = 0: Break signal is not distributed to low threshold event.<br>LTBSD[i] = 1: Break signal i is distributed to low threshold event. |

### Filter y threshold monitor status register (HPDF_FLTyTMSTAT)

Address offset: 0x128 + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

Note: All the bits of HPDF_FLTyTMSTAT are automatically reset when FLTEN=0.

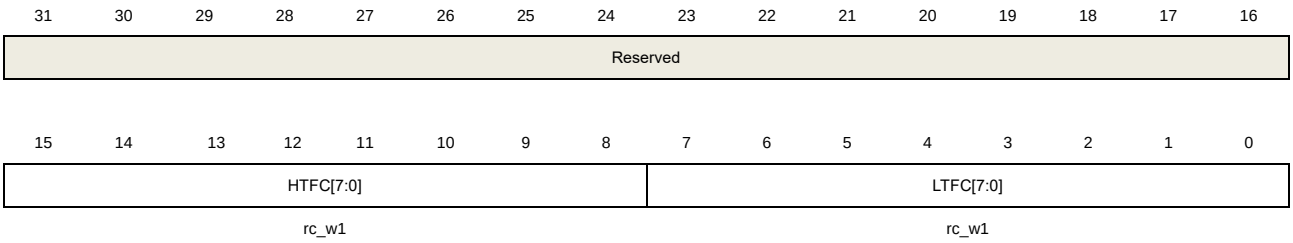



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | HTF[7:0] | Threshold monitor high threshold flag<br>HTF[x]=0: No high threshold error on channel x<br>HTF[x]=1: High threshold error on channel x<br>It is set by hardware. It can be cleared by software setting the corresponding HTFC[7:0] bit in the HPDF_FLTyTMFC register. |
| 7:0 | LTF[7:0] | Threshold monitor low threshold flag<br>LTF[x]=1: No low threshold error on channel x<br>LTF[x]=1: Low threshold error on channel x<br>It is set by hardware. It can be cleared by software setting the corresponding LTFC[7:0] bit in the HPDF_FLTyTMFC register. |

### Filter y threshold monitor flag clear register (HPDF_FLTyTMFC)

Address offset: 0x12C + 0x80 * y, (y = 0, 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



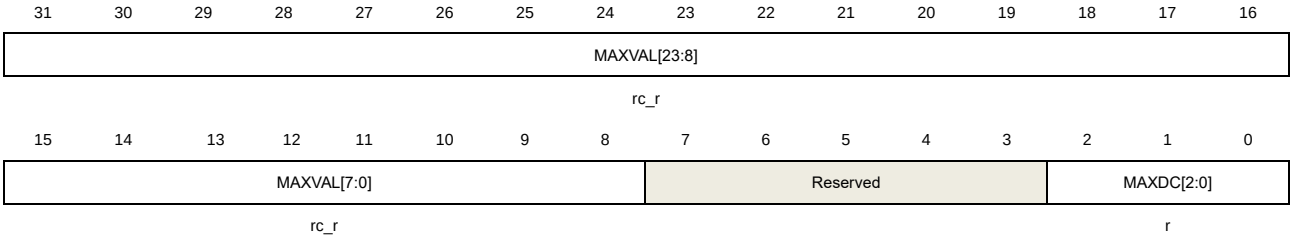
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | HTFC[7:0] | Clear the threshold monitor high threshold flag<br>HTFC[y]=0: No effect<br>HTFC[y]=1: Clear the threshold monitor high threshold flag on channel x |
| 7:0 | LTFC[7:0] | Clear the threshold monitor low threshold flag<br>LTFC[y]=0: No effect<br>LTFC[y]=1: Clear the threshold monitor low threshold flag on channel x |

### Filter y extremes monitor maximum register (HPDF_FLTyEMMAX)

Address offset: 0x130 + 0x80 * y, (y = 0, 3)

Reset value: 0x8000 0000

This register has to be accessed by word (32-bit).



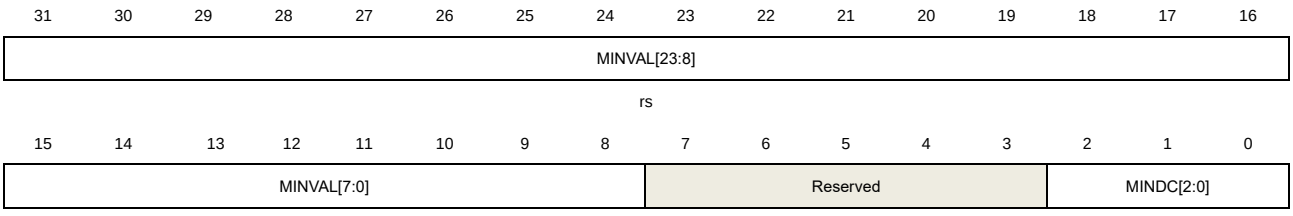

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | MAXVAL[23:0] | Extremes monitor maximum value<br>These bits are set by hardware and indicate the highest value of channel converted by HPDF_FLTy.<br>These bits can be reset by reading of this register. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | MAXDC[2:0] | Extremes monitor maximum data channel.<br>This bits indicate the channel on which the data is stored into MAXVAL[23:0]. It can be cleared by reading of this register. |

### Filter y extremes monitor minimum register (HPDF_FLTyEMMIN)

Address offset: 0x134 + 0x80 * y, (y = 0, 3)

Reset value: 0x7FFF FF00

This register has to be accessed by word (32-bit).



rs rc_r

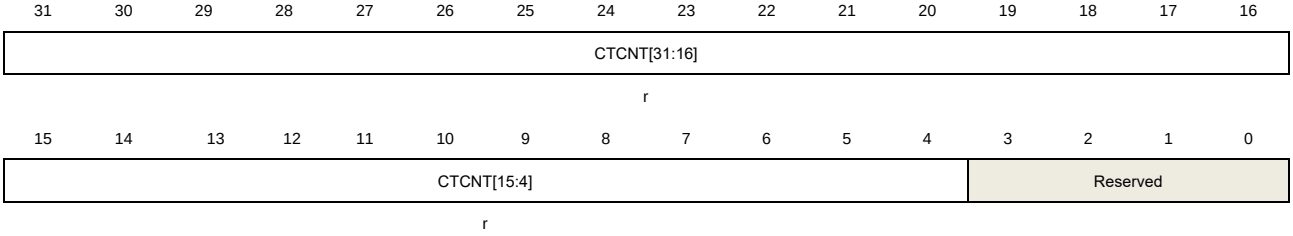
| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | MINVAL[23:0] | Extremes monitor minimum value<br>These bits are set by hardware and indicate the lowest value converted by HPDF_FLTy.<br>These bits can be reset by reading of this register. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | MINDC[2:0] | Extremes monitor minimum data channel<br>This bit indicate the channel on which the data is stored into MINVAL[23:0]. It can be cleared by reading of this register. |

### Filter y conversion timer register (HPDF_FLTyCT)

Address offset: 0x138 + 0x80 * y, (y = 0 to 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | CTCNT[27:0] | conversion time counted by HPDFCLK<br>t = CNVCNT[27:0] / $f_{HPDFCLK}$<br>The timer has an input clock (system clock $f_{HPDFCLK}$) from the HPDF clock. The conversion time measurement starts at the beginning of each conversion and stops when the conversion is complete (the interval between the first and last serial sample). Conversion time measurement stops and CNVCNT[27:0] = 0 only when the filter is bypassed (FOSR[9:0] = 0). The times counted are:<br>if FAST=0 (or first conversion in continuous mode if FAST=1):<br>for Sincx filters: t = [SFOR * (IOR-1 + FORD) + SFO] / $f_{CKIN}$.<br>for FastSinc filter: t = [SFOR * (IOR-1 + 4) + 2] / $f_{CKIN}$.<br>if FAST=1 in continuous mode (except first conversion):<br>t = [SFOR * IOR] / $f_{CKIN}$.<br>in case if FOSR = SFOR [9:0]+1 = 1: CNVCNT = 0 (counting is stopped, conversion time: t = IOR / $f_{CKIN}$)<br>where $f_{CKIN}$ is the channel input clock frequency (on a given channel CKINx pin) or the input data rate in the case of parallel data input (from CPU/DMA writes)<br>**Note**: The timer will also counts this interrupt time when a conversion is interrupted |

| | | |
|---|---|---|
| | | (for example by disabling/enabling the selected channel). |
| 3:0 | Reserved | Must be kept at reset value. |

# 46. Real-time decryption (RTDEC) (For GD32H737_757_759)

## 46.1. Overview

The real-time decryption (RTDEC) allows to decrypt in real-time according to information of the read request address. RTDEC can configure four independent and different encrypted areas. And each area has the option of execute-only or execute-never enforcement to choose.

For a good real-time performance, RTDEC uses the counter (CTR) mode of AES-128. Since RTDEC using AES-128 in counter mode, the whole area has to be re-encrypted with an updated cryptographic context (key or initialization vector) when the data or code of one encrypted area is changed. This feature makes RTDEC only suitable for decrypting read-only content, like that stored in external flash.

**Note**: It is necessary to access the external read-only memory using the memory map mode when RTDEC is used with Octal-SPI (OSPI). It is important to know that CPU memories and RTDEC follow little endian notation, and AES hardware accelerator follows big endian notation.

## 46.2. Characteristics

- Software configurable encrypted areas up to 4.
- Granularity is 4096 bytes in RTDEC programmed areas.
- Area configuration can be locked.
- Optional decryption modes: execute-only or execute-never.
- Every area can be configured the independent 128-bits key, 16-bits area firmware version, and 64-bits application-defined nonce. If the user performed an encryption, one or more of those should be modified.
- Confidentiality and completeness protection for encryption keys.
  - 128-bits key registers are write-only, with software locking mechanism
  - 8-bits CRC is calculated automatically by hardware, and it's used as the public key information
- The real-time decryption when OSPI memory-mapped read operations.
  - Use of AES-128 in CTR mode
  - Support key stream FIFO with depth 4
  - Support various read size
  - Decryption / encryption with physical address of the reads
- Support for GD32 OSPI pre-fetching mechanism.

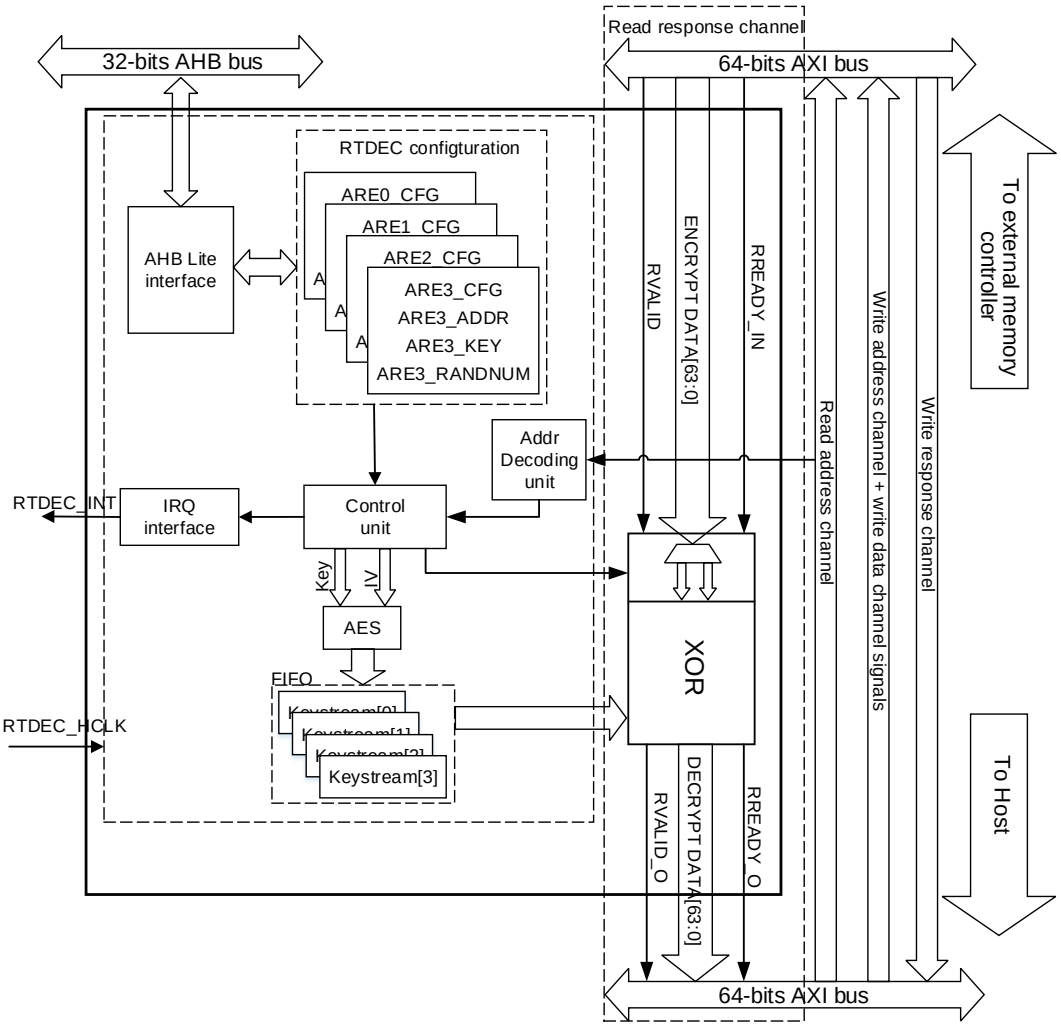
## 46.3. Function overview

The RTDEC (real-time decryption) module allows real-time decryption of the Arm® AXI or AHB traffic based on the read request address information. Original purpose of RTDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR Flash devices. The RTDEC performs real-time decryption during OSPI memory-mapped read operation.

### 46.3.1. RTDEC real-time decryption introduction

*Figure 46-1. RTDEC block diagram (for RTDEC0 and RTDEC1)* shows the block diagram of the RTDEC.

**Figure 46-1. RTDEC block diagram (for RTDEC0 and RTDEC1)**




**Note**: RTDEC_HCLK is RTDEC clock input from AHB clock, RTDEC_INT is RTDEC global interrupt.

### 46.3.2. RTDEC real-time decryption introduction

#### RTDEC typical usage

RTDEC fundamental purpose is to protect the secrecy of the execute-only code. The code can be the execute-only firmware library is executed from external flash memory. RTDEC can also protect “code + data library” (which is read only), and data library (which is read only and execute-never) stored in external flash memory.

In order to protect the completeness of the decryption keys and also to protect other RTDEC configurations against software denial of services attacks, RTDEC provides the configuration register locking mechanism.

It is necessary to access the external read-only memory using the memory map mode when RTDEC is used with OSPI.

#### RTDEC principle

The principle of RTDEC is to analyze all read address channel transactions between the MCU and a peripheral, the read address channel transactions is on the AXI interconnect bus, like the OSPI controller.

The control logic will trigger a key stream computation if the read request is within one of the four areas programmed in RTDEC. The key stream is computed by RTDEC using AES-128 in counter mode. Any accesses outside the RTDEC programmed areas belong to a non-encrypted area. Granularity for the area determination is 4KB. That is, size of the RTDEC programmed areas is an integer multiple of 4KB.

Each RTDEC areas can be programmed by registers AREx_CFG, AREx_SADDR, AREx_EADDR, AREx_NONCE and AREx_KEY, where x=0 to 3. It is could be configured whether the content of the RTDEC area is code (execute-only), data (execute-never) or both through MODE[1:0] (in the register AREx_CFG).

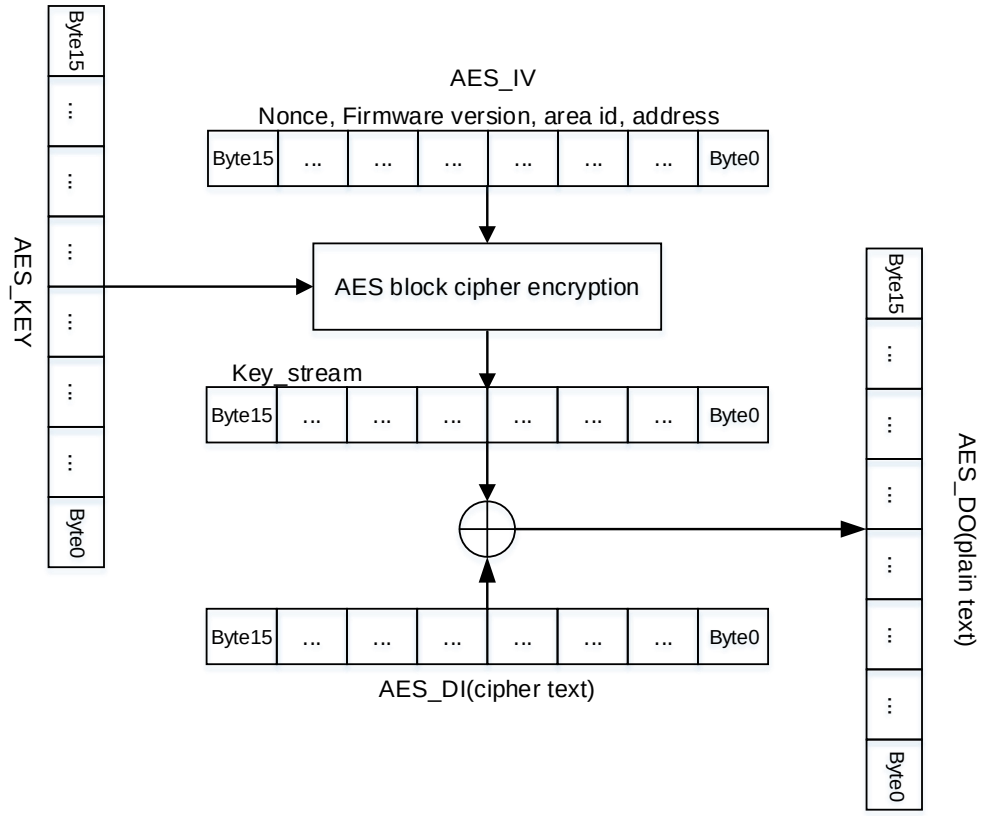
**Note:** RTDEC does not prevent overlapping of areas but it is an invalid programming, and the user should try to avoid areas overlapping in application software.

When RTDEC decrypts incremental or wrap bursts they should not cross the 4096-byte aligned address boundaries. RTDEC cannot decrypt part data of an AXI transaction.

### 46.3.3. Decryption with AES-128 in counter mode

*Figure 46-2. Decryption flow with AES-128 in CTR mode* shows RTDEC how to use Advanced Encryption Standard (AES) algorithm in counter mode. AES is announced by Federal Information Processing Standards Publication 197, November 26, 2001. And it is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

**Figure 46-2. Decryption flow with AES-128 in CTR mode**




During decryption, a special key steam information will be computed by using AES block cipher for every 128-bit data block. The AES_IV and AES_KEY are defined as below:

1. The initialization vector AES_IV[127:0] = RTDEC_AREx_NONCE[63:0] || 16'b 0000 0000 0000 0000 || RTDEC_AREx_CFG[31:16] || 2'b 00 || x || ReadAddress[31:4].
2. The key material AES_KEY[127:0] = RTDEC_AREx_KEY3[31:0] || RTDEC_AREx_KEY2[31:0] || RTDEC_AREx_KEY1[31:0] || RTDEC_AREx_KEY0[31:0].

**Note:** Above x (x = 2'b 00 to 2'b 11) is the Area ID of the selected encrypted area.

After the 128-bit key stream is computed, the 128-bit clear text is produced by 128-bit key stream XORed with 128-bit cipher text data.

The 128-bit AES_DI and AES_DO data blocks are defined following the rule below: AES_Dx[127:0] = AXI_word(@+0x8)[63:0] || AXI_word(@)[63:0]. Where @ is the hexadecimal address, and it is used to compute the key stream.

When the read request is within a non-encrypted area, or in an encrypted area but the decryption is not enabled, the 128-bit AXI data is not changed.

## 46.3.4. RTDEC configuration

### RTDEC initialization process

The trusted initialization of RTDEC registers is an important aspect when use RTDEC.

Because it involves secret keys initialization and critical options like MODE[1:0] bits initialization.

There are two solutions of RTDEC trusted initialization are introduced here below.

**Note:** Those sequences are suggested for production code. Because it is not always recommended to set bit ARE_K_LK or ARE_CFG_LK (lock the key or the area configuration) during code development.

Writes to area configuration registers are effective when the bit ARE_CFG_LK is 0, even if the bit ARE_EN is 1.

**One key for all areas initialization solution**

In this solution, the secret key used to decrypt the RTDEC four protected areas can belong to one entity.

The follow configuration sequence of RTDEC is recommended:

- Write the correct MODE[1:0] value in RTDEC_AREx_CFG register (where x=0 to 3).
- Program RTDEC_AREx_KEY0...3 registers (where x=0 to 3) using the sequence described in ARE_K_CRC (make sure to have a valid CRC). Note that key registers are write only, read from them is illegal.
- Check the key CRC. If it's ok, set bit ARE_K_LK in RTDEC_AREx_CFG register (where x=0 to 3). ARE_K_LK bit cannot be cleared, that is the key registers in this area x are no more changed.
- While you have an area x to decrypt do it. It is not necessarily have to be performed by the entity (that owns the decryption keys) when this work does.
  - Check if the key CRC matches to the encrypted binary stored in the area.
  - Configure the detailed information matching to this binary. The information are nonce, firmware version, start and end address.
  - Set bit ARE_EN in RTDEC_AREx_CFG (where x=0 to 3) to enable decryption of this area.
  - Set bit ARE_CFG_LK in RTDEC_AREx_CFG (where x=0 to 3). This bit cannot be cleared, that is the area configuration is no more changed.

**Note:** For a given area, the key registers and associated CRC are cleared by hardware when MODE[1:0] bits are changed. So MODE[1:0] bits must be written firstly before other operating, and after programming RTDEC_AREx_KEY0...3 (where x=0 to 3) registers the MODE[1:0] bits must not be modified.

**One key for one area initialization solution**

In this solution, the secret key used to decrypt the RTDEC one (or more) protected areas can belong to one entity.

The follow configuration sequence of RTDEC is recommended.

While you have an area x to decrypt do it, such as x is 0. It is have to be performed by the entity (that owns the corresponding decryption key) when this work does.

- Write the correct MODE[1:0] value in RTDEC_ARE0_CFG register.
- Program RTDEC_ARE0_KEY0...3 registers using the sequence described in ARE_K_CRC (read from area configuration register to have a valid CRC). Note that key registers are write only, read from them is illegal.
- Check the key CRC. If it's ok, set bit ARE_K_LK in RTDEC_ARE0_CFG register. This bit cannot be cleared, that is the key registers are no more changed.
- Configure the detailed information matching to the protected firmware. The information are nonce, firmware version, start and end address.
- Set bit ARE_EN in RTDEC_ARE0_CFG to enable decryption of this area.
- Set bit ARE_CFG_LK in RTDEC_ARE0_CFG. This bit cannot be cleared, that is the area configuration is no more changed.

**Note:** For a given area, the key registers and associated CRC are cleared by hardware when MODE[1:0] bits are changed. So MODE[1:0] bits must be written firstly before other operating, and after programming RTDEC_ARE0_KEY0...3 registers the MODE[1:0] bits must not be modified.

#### Reset of RTDEC

The correct key loading sequence described in ***RTDEC initialization process*** must be performed when each time RTDEC is reset (in this case, bit ARE_K_CRC of RTDEC_AREx_CFG is 0). When the RTDEC is reset by hardware we suggested that user's application software to guarantee the correct key loading sequence.

### 46.3.5. RTDEC error management

RTDEC could manages below errors automatically,

- Illegal read to RTDEC_AREx_KEY0...3 registers. Where x is 0 to 3.
- While bit ARE_CFG_LK or ARE_K_LK in RTDEC_AREx_CFG is set, illegal write to RTDEC_AREx_KEY0...3 registers. Where x is 0 to 3.
- While bit ARE_CFG_LK in RTDEC_AREx_CFG is set, illegal write to RTDEC_AREx_CFG, RTDEC_AREx_SADDR, RTDEC_AREx_EADDR or RTDEC_AREx_NONCE0/1 registers. Where x is 0 to 3.
- While MODE[1:0] is 2'b00, illegal read to an execute-only area. While MODE[1:0] is 2'b01, illegal execution request to an execute-never area. These illegal requests return 0, without bus error.
- Key error: When an abort event happened, the key register will be cleared. At this time if there are some read requests, these requests will return 0, without bus error. The abort event maybe tamper detection, unauthorized debug connection, untrusted boot or SPC level low to no protection demotion.

If the bit SECIE, ECONEIE or KEIE is set, one or more of above errors occurs will generate an interrupt. For details refer to ***RTDEC interrupts***.

**Note**: After a key error occurs, a system reset must be performed first, followed by reconfiguring the RTDEC, in order for the RTDEC to function normally again.

## 46.4. RTDEC interrupts

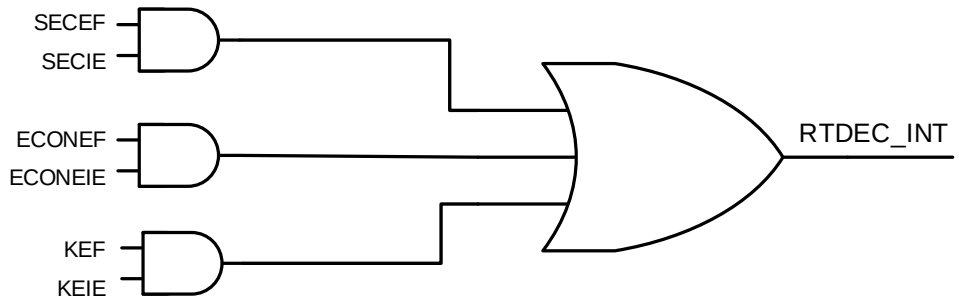
RTDEC can generate three independent maskable interrupt events, the interrupt flag is set after following security events:

- Illegal read or write access to keys (SECEF flag), refer to ***Interrupt flag register (RTDEC_INTF)***.
- Illegal write to an area's configuration register while ARE_CFG_LK=1 (SECEF flag), refer to ***Interrupt flag register (RTDEC_INTF)***.
- While MODE[1:0] is 2'b00, read access to an execute-only area. While MODE[1:0] is 2'b01, execute access to a execute-never data area. Both are triggering the ECONEF flag.
- Key error (encrypted area read as zero), triggering the KEF flag. Refer to ***Interrupt flag register (RTDEC_INTF)***.

All of the interrupt events are ORed together before being sent to the interrupt controller, so the RTDEC can only generate a single interrupt request to the controller at any given time.

Below block diagram illustrates the RTDEC module interrupt connection

**Figure 46-3. RTDEC interrupt mapping diagram**




By setting corresponding SECIE, ECONEIE or KEIE bits in RTDEC_INTEN register, you can enable or disable RTDEC interrupt sources. It is shown as ***Table 46-1. RTDEC interrupt requests***. Status of the interrupt event is found in RTDEC_INTF register, and the interrupt flag can be cleared using RTDEC_INTC register.

**Table 46-1. RTDEC interrupt requests**

| Interrupt event | Event flag | Enable control bit |
|---|---|---|
| Security error | SECEF | SECIE |
| Execute-only, Execute-never error | ECONEF | ECONEIE |
| Key error | KEF | KEIE |

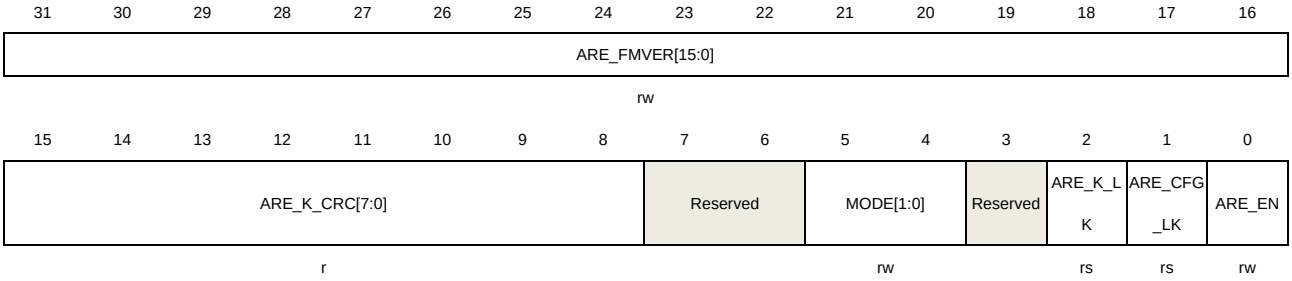
## 46.5. Registers definition

RTDEC0 base address: 0x5200 B800
RTDEC1 base address: 0x5200 BC00

### 46.5.1. Area x configuration register (RTDEC_AREx_CFG)

Address offset: 0x20 + 0x30 * x (x = 0 to 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | ARE_FMVER[15:0] | Area firmware version bits<br>Before enabling the ARE_EN bit in the RTDEC_AREx_CFG register, the area firmware version must be configured first. |
| 15:8 | ARE_K_CRC[7:0] | 8-bit CRC of area key bits<br>As ARE_K_LK = 0, if user loads the key of the area in this exact sequence: KEY0 -> KEY1 -> KEY2 -> KEY3, the ARE_K_CRC[7:0] will automatically computed by hardware. When a new valid sequence is initiated, a new computation starts immediately, and ARE_K_CRC[7:0] is read as 0 until a valid sequence is completed.<br>When ARE_K_LK=1, ARE_K_CRC remains unchanged until the next reset.<br>The standard CRC-8-CCITT algorithm X8 + X2 + X + 1 is used for CRC calculation, the calculation result is an 8-bit checksum.<br>These bits are read only.<br>**Note:** When the last bit of the key has been written, CRC information is updated. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5:4 | MODE[1:0] | RTDEC mode bits<br>These bits configure the RTDEC operating mode for this area:<br>00: Only decrypt code accesses.<br>01: Only decrypt data accesses.<br>10: decrypt all read accesses (code or data).<br>11: Reserved.<br>When the MODE[1:0] bits are changed, the area's key and associated CRC are |

| | | |
|---|---|---|
| | | automatically cleared. |
| 3 | Reserved | Must be kept at reset value. |
| 2 | ARE_K_LK | Area key lock bit<br>0: RTDEC_AREx_KEY registers write is enabled.<br>1: RTDEC_AREx_KEY registers write is disabled. ARE_K_CRC[7:0] is also locked.<br>This bit is set by software only once. It cannot be cleared until a RTDEC reset. When this bit is set, it disables write access to this area key registers. |
| 1 | ARE_CFG_LK | Area configure lock bit<br>0: RTDEC_AREx_CFG, RTDEC_AREx_SADDR, RTDEC_AREx_EADDR, RTDEC_AREx_NONCEy (y = 0, 1), RTDEC_AREx_KEYy (y = 0 to 3) registers write is enabled.<br>1: RTDEC_AREx_CFG, RTDEC_AREx_SADDR, RTDEC_AREx_EADDR, RTDEC_AREx_NONCE, RTDEC_AREx_KEY registers write is disabled.<br>This bit is set by software only once. It cannot be cleared until a RTDEC reset. When this bit is set ARE_K_LK bit is forced to 1, and it disables write access to this area configuration, start address, end address and nonce registers. |
| 0 | ARE_EN | Area real-time decryption enable bit<br>0: Area real-time decryption is disabled.<br>1: Area real-time decryption is enabled.<br>The area configure information must be valid before this bit is set. |

### 46.5.2. Area x start address register (RTDEC_AREx_SADDR)

Address offset: 0x24 + 0x30 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_SADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ARE_SADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_SADDR[31:0] | Area x AXI start address bits<br>These bits should be written before the corresponding ARE_EN is set in the RTDEC_AREx_CFG register.<br>Writing these bits is invalid if the ARE_CFG_LK bit is set in the RTDEC_AREx_CFG register.<br>**Note:** The first 12 bits (LSB) and the last 4 bits (MSB) are ignored when determining the area. |

The 4 MSB bits and the 12 LSB bits return 0, when read this register.

### 46.5.3. Area x end address register (RTDEC_AREx_EADDR)

Address offset: 0x28 + 0x30 * x (x = 0 to 3)
Reset value: 0x0000 0FFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_EADDR[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_EADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_EADDR[31:0] | Area x AXI end address bits<br>These bits should be written before the corresponding ARE_EN is set in the RTDEC_AREx_CFG register. And ARE_EADDR cannot be less than ARE_SADDR.<br>Writing these bits is invalid if the ARE_CFG_LK bit is set in the RTDEC_AREx_CFG register.<br>**Note:** The first 12 bits (LSB) and the last 4 bits (MSB) are ignored when determining the area.<br>The 4 MSB bits return 0 and the 12 LSB bits return 1, when read this register. |

### 46.5.4. Area x nonce register 0 (RTDEC_AREx_NONCE0)

Address offset: 0x2C + 0x30 * x (x = 0 to 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_NONCE0[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_NONCE0[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_NONCE0[31:0] | Area x nonce bits, ARE_NONCE[31:0].<br>These bits should be written before the corresponding ARE_EN bit is set in the RTDEC_AREx_CFG register.<br>Writing these bits is invalid if the ARE_CFG_LK bit is set in the |

RTDEC_AREx_CFG register.

### 46.5.5. Area x nonce register 1 (RTDEC_AREx_NONCE1)

Address offset: 0x30 + 0x30 * x (x = 0 to 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_NONCE1[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_NONCE1[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_NONCE1[31:0] | Area x nonce bits, ARE_NONCE[63:32].<br>Refer to ARE_NONCE0[31:0] description |

### 46.5.6. Area x key register 0 (RTDEC_AREx_KEY0)

Address offset: 0x34 + 0x30 * x (x = 0 to 3)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_KEY0[31:16] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ARE_KEY0[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |

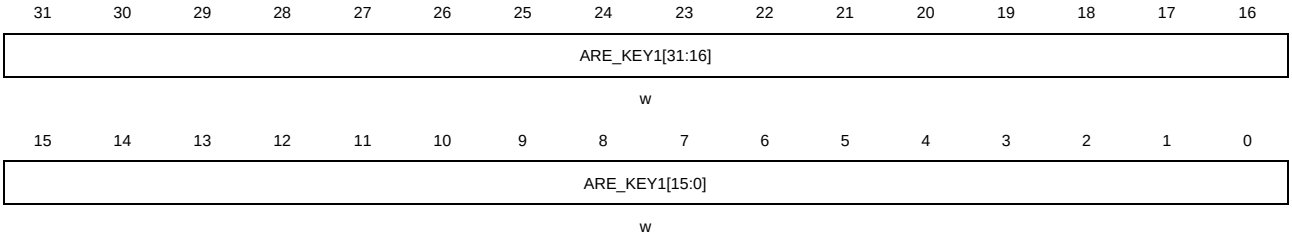
| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_KEY0[31:0] | Area x key bits, ARE_KEY[31:0].<br>These bits should be written before the corresponding ARE_EN bit is set in the RTDEC_AREx_CFG register.<br>Writing these bits is invalid if the ARE_CFG_LK bit or ARE_K_LK bit is set in the RTDEC_AREx_CFG register. When read these bits it returns 0.<br>**Note:** RTDEC_AREx_KEY0 register and associated ARE_K_CRC are erased when application successfully changes bits MODE[1:0] in RTDEC_AREx_CFG register. |

### 46.5.7. Area x key register 1 (RTDEC_AREx_KEY1)

Address offset: 0x38 + 0x30 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



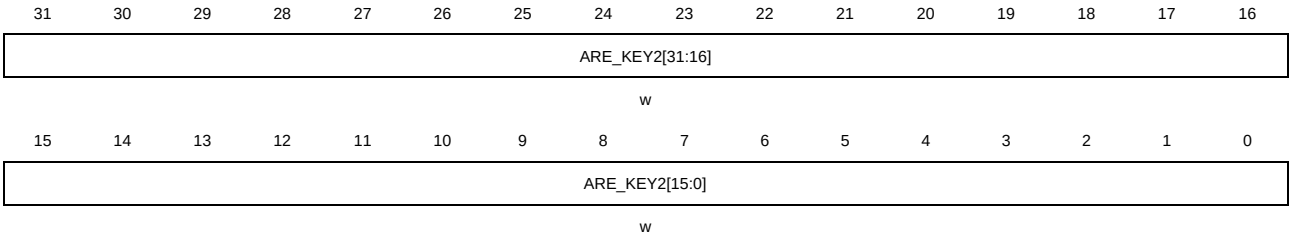

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_KEY1[31:0] | Area x key bits, ARE_KEY[63:32].<br>Refer to RTDEC_ARE_KEY0[31:0] description |

### 46.5.8. Area x key register 2 (RTDEC_AREx_KEY2)

Address offset: 0x3C + 0x30 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



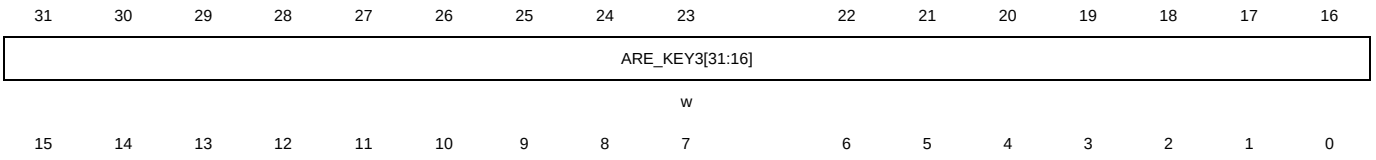

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_KEY2[31:0] | Area x key bits, ARE_KEY[95:64].<br>Refer to RTDEC_ARE_KEY0[31:0] description |

### 46.5.9. Area x key register 3 (RTDEC_AREx_KEY3)

Address offset: 0x40 + 0x30 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



| ARE_KEY3[15:0] |
|---|
| w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | ARE_KEY3[31:0] | Area x key bits, ARE_KEY[127:96].<br>Refer to RTDEC_ARE_KEY0[31:0] description |

## 46.5.10. Interrupt flag register (RTDEC_INTF)

Address offset: 0x300

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | KEF | ECONEF | SECEF |
| | | | | | | | | | | | | | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | KEF | Key error interrupt flag<br>0: No key error is detected.<br>1: Key error is detected. Read access detected on an enabled encrypted area after an abort event. An interrupt will occur if the KEIE bit is set in RTDEC_INTEN. This bit is set by hardware, when a read access occurs on encrypted or dencrypted area following the reset of the key registers by an abort event (tamper detection, unauthorized debugger connection, untrusted boot, SPC level low to no protection demotion).<br>Cleared by writing 1 to the KEC bit in the RTDEC_INTC register.<br>Any subsequent read to encrypted or dencrypted area returns 0 after KEF bit is set. It remains until RTDEC keys are initialized again. |
| 1 | ECONEF | Execute-only or execute-never error interrupt flag<br>0: No execute-only error or execute-never error is detected.<br>1: Execute-only error or execute-never error is detected. Read access detected on an area with MODE[1:0] set to 00, or execute access detected on an area with MODE [1:0] set to 01. An interrupt will occur if bit ECONEIE is set in RTDEC_INTEN register.<br>Set by hardware, when a read access but not an instruction fetch is detected on any encrypted area with MODE[1:0] set to 00. It is also set when an instruction fetch but not a read access is detected on any encrypted area with MODE[1:0] set to 01. |

| | | |
|---|---|---|
| | | Cleared by writing 1 to the ECONEC bit in the RTDEC_INTC register.<br>If the access is illegal, RTDEC returns 0. |
| 0 | SECEF | Security error interrupt flag<br>0: No security error is detected.<br>1: Security error is detected. An interrupt will occur if SECEIE bit is set in RTDEC_INTEN register.<br>Set by hardware, when at least one security error has been detected (illegal access to keys, illegal write on locked configuration).<br>Cleared by writing 1 to SECEC bit in the RTDEC_INTC register. |

## 46.5.11. Interrupt flag clear register (RTDEC_INTC)

Address offset: 0x304
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | KEC | ECONEC | SECEC |
| | | | | | | | | | | | | | w | w | w |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | KEC | Key error flag clear<br>0: No effect<br>1: Clear key error flag<br>**Note:** To be able to read or execute again any encrypted region, RTDEC key registers should properly reinitialized but not just clear KEF. |
| 1 | ECONEC | Execute-only or execute-never error flag clear<br>0: No effect<br>1: Clear execute-only or execute-never error flag |
| 0 | SECEC | Security error interrupt flag<br>0: No effect<br>1: Clear security error flag |

## 46.5.12. Interrupt enable register (RTDEC_INTEN)

Address offset: 0x308
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | KEIE | ECONEIE | SECEIE |
| | | | | | | | | | | | | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | KEIE | Enable bit for key error interrupt<br>Software set and cleared<br>0: Disable key error interrupt<br>1: Enable key error interrupt |
| 1 | ECONEIE | Enable bit for execute-only or execute-never error interrupt<br>Software set and cleared<br>0: Disable execute-only or execute-never error interrupt<br>1: Enable execute-only or execute-never error interrupt |
| 0 | SECEIE | Enable bit for security error interrupt<br>Software set and cleared<br>0: Disable security error interrupt<br>1: Enable security error interrupt |

# 47. Filter arithmetic accelerator (FAC)

## 47.1. Overview

The filter arithmetic accelerator unit consist of multiplier, accumulator and address generation logic, so as to index vector elements stored in local memory. Circular buffering is valid for both input and output, which allows to realize finite impulse response (FIR) filters and infinite impulse response (IIR) filters. The unit support CPU to be free from frequent or lengthy filtering operations, compared with software implementation, it can accelerate calculations and the processing speed of time critical tasks.

## 47.2. Characteristics

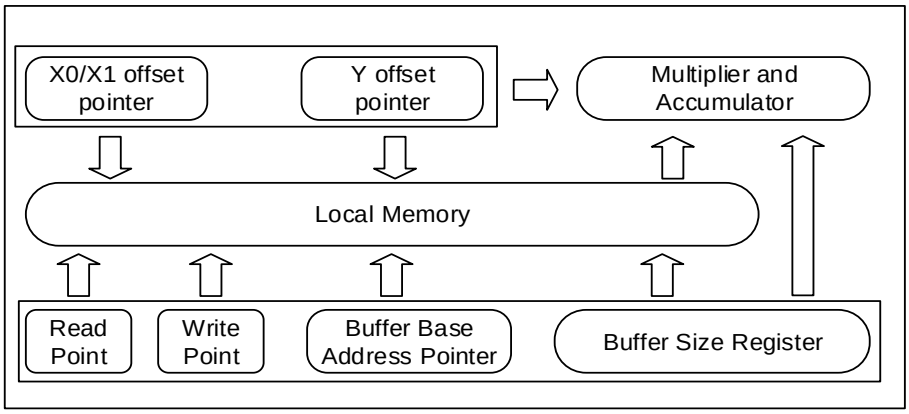
- Fixed or float multiplier and accumulator.
- 256 x 32-bit local memory.
- 16-bit fixed-point or 32-bit float point input and output.
- Up to three buffers, two input buffers and one output buffer.
- Buffer can be circular.
- FIR and IIR can be realized.
- Vector functions support convolution, Dot product, correlation functions.
- Data can be read and written through DMA.

## 47.3. Function overview

### 47.3.1. General description

The unit can be configured based on fixed point multiplier and accumulator or float point multiplier and accumulator. Two 16-bit input signed data or 32-bit input float data from local memory are taken to MAC, they are multiplied together and added to the accumulator. A set of pointers, which could be incremented, decremented and loaded or reset by hardware, index the address of the input data in memory. Built-in sequencer arranges control the pointer and MAC operations to perform the requested operation in order.

**Figure 47-1 FAC structure diagram**




The processor or DMA controller load two input vectors into the local memory to calculate a dot product, select and start the requested operation. The elements of input vector is extracted from memory, multiplied together and then accumulated the multiplier output together. After processing the requested operation, the local memory is used to store the contents of the accumulator, the processor or DMA can access the corresponding address.

The finite impulse response filter operation repeatedly calculates the dot product, which refer to the coefficient data and a input sample data, along with discarding the least recent sample and adding a new sample.

The IIR filter calculates the product of the feedback coefficient and the previous output value, and adds the calculated result to the FIR convolution result to obtain the final filtered output.

### 47.3.2. Local memory and buffers

The unit contains 256 x 32 bit memory which can be read and write. X0 buffer and X1 buffer save the input values,and Y buffer saves the output values.

The locations of the buffers are specific as follows: x0_base, the X0 buffer base address, x1_base, the X1 buffer base address, y_base, the Y buffer base address.

The sizes of the buffers are specific as follows: x0_buf_size, the number of word allocated to the X0 buffer, x1_buf_size, the number of word addresses allocated to the X1 buffer, y_buf_size, the number of word addresses allocated to the Y buffer. Above parameters could be configured in corresponding register.

Through using the initialization functions, the X0 buffer, X1 buffer and Y buffer can be initialized. The data is transferred to the target buffer, which is indicated by a write pointer. The write pointer increase along with each new write, if the pointer arrives the end of the buffer space, the pointer returns to the base address. Thus, the vector element is loaded before operation, it can also be used to load the filter coefficients and initialize the filter.

Buffer configuration registers configure the buffer sizes and base address. The filter function specifies required buffer size for each function, while the base address in the local memory could be configured optionally, therefore, considering that all buffers address from 0x00 to 0xFF, in other words, base address add buffer size should be less than 256. The location and

size of the buffers lack of constraint, even they can overlap completely. Do not overlap the buffer of filter function to avoid abnormal operation.

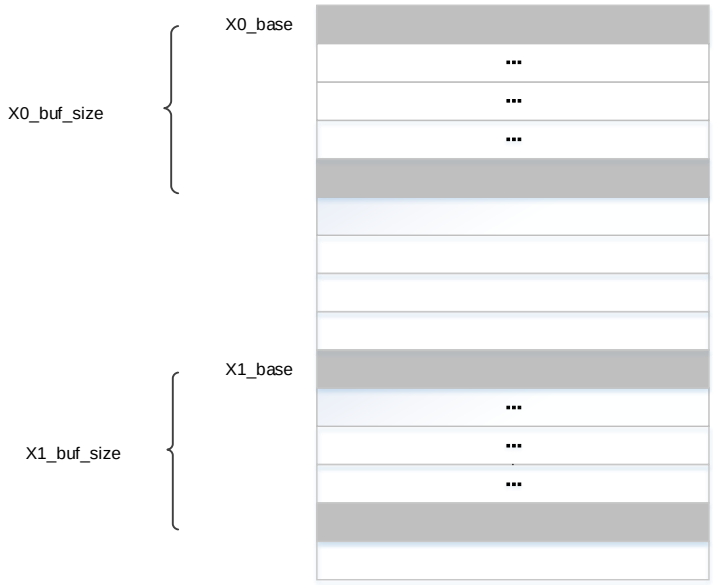
An optional headroom is added to the buffer size if circular buffer operation is required, Moreover, for regulating the DMA or CPU activity, it is necessary to set watermark level. The headroom value and watermark level should be selected according to the actual application.

Usually, for more data throughput, the input buffer always has data, the headroom is slightly greater than the watermark level, so as to allow interrupt or DMA latency. On the other side, if the input data providing speed is less than the unit processing speed, the input buffer could be empty and wait for writing the next data. Therefore, the watermark level can be equal to headroom, so as to ensure that the input does not overflow.

### 47.3.3. Input buffers

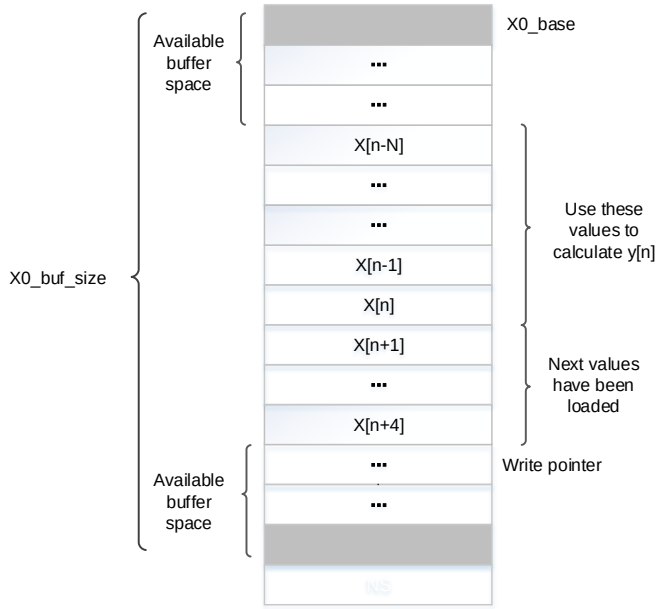
X0 buffer and X1 buffer store the input data of MAC, every multiplication operation multiplies two data together, one data from X0 buffer and the other from X1 buffer. The read address offset is generated by the pointer of the control unit, which is relative to the buffer base address.

**Figure 47-2 Input buffer area**




If the X0 buffer works in circular mode, new data will continue to be transferred to the input buffer in case that space is available. For digital filters, preloading the buffer is optional. When the operation is started, if no input samples are written to X0 buffer, the buffer is flagged as empty. DMA or CPU are required to load new samples, the request will not disappear until there are enough samples to begin operation.

**Figure 47-3 Circular input buffer area**




The X1 buffer can only work as a not circular buffer. Unless the contents of the buffer do not follow operation change, X1 buffer usually needs to be pre-loaded. In addition, X1 buffer could store the filter coefficients for filter functions.

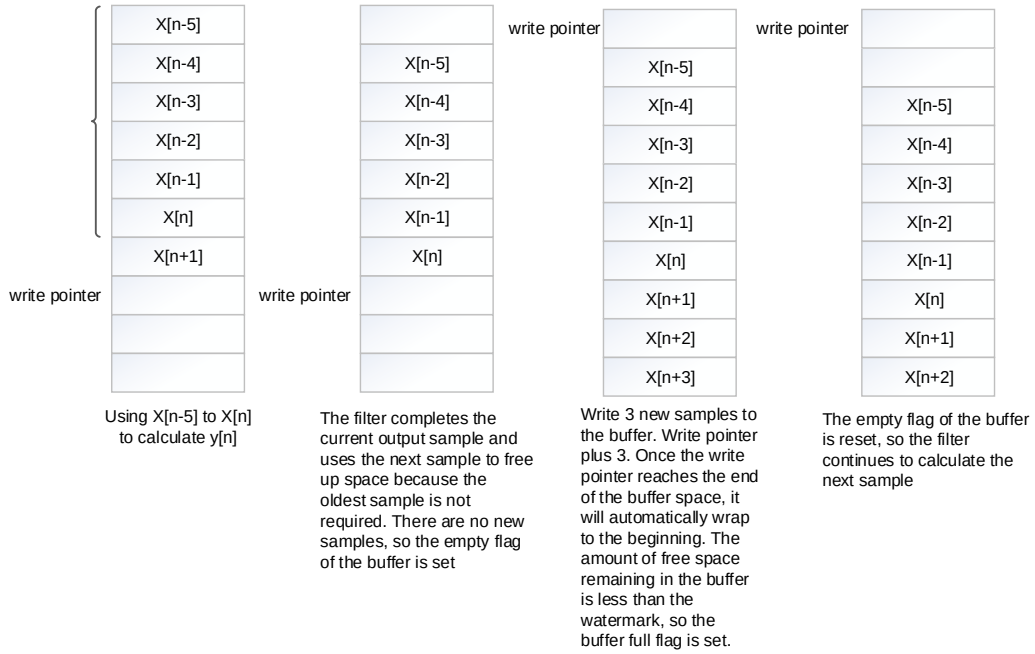
For a circular buffer, the allocated space of the buffer (x0_buf_size) should be greater than element numbers in use, therefore, new values are always available in the buffer. ***Figure 47-3 Circular input buffer*** show the buffer layout for a filter operation. For calculating an output sample y[n], a set of input samples is called, from x[n-N] to x[n], length is N+1. Once calculating is finished, call input samples x[n-N+1] to x[n+1], and then start calculating y[n+1]. The least-recent input sample (x[n-N]) is discarded, and a new sample(x[n+1]) is added.

It is necessary to ensure that the new sample x[n+1] in the buffer space is available if required. If x[n+1] is not available, the execution will be suspended and the buffer is flagged as empty, unless a new sample is added. If a timer or other peripheral control the flow of samples, considering that the source provide it is slower than the filter sample processing, the buffer work in empty states usually.

The watermark threshold is configured in the X0_WBFF bit field of the FAC_X0BCFG register, If the amount of free space in X0 buffer is less than the watermark threshold, the X0 buffer is regarded as full state. Interrupts are generated if the full flag is not set, more data are requested for the buffer while FAC in enable,. Under one interrupt, the watermark permit that transferring several data without considering overflow risk. However, the OFEF error flag is set if overflow occur, on the same time, the write data is ignored and the write pointer is not incremented.

***Figure 47-4 Circular input buffer operation*** shows the change process of X0 buffer during an 6-tap FIR filtering processing, while the watermark is set to 3.

**Figure 47-4 Circular input buffer operation**

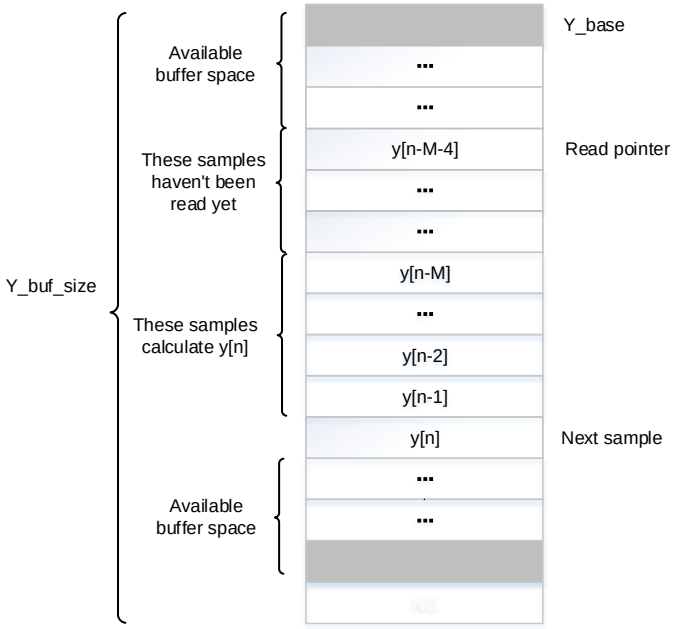



## 47.3.4. Output buffer

The output of an accumulation is held in Y buffer, the buffer space will be released when the output value is read by processor or DMA.

The read pointer points to the address where the data needs to be read during the read operation, the read data is taken out from the read pointer address when a read command occurs, meanwhile the read pointer is incremented. While pointer reaches the end of the Y buffer space, it returns back to base address.

**Figure 47-5 Circular output buffer**



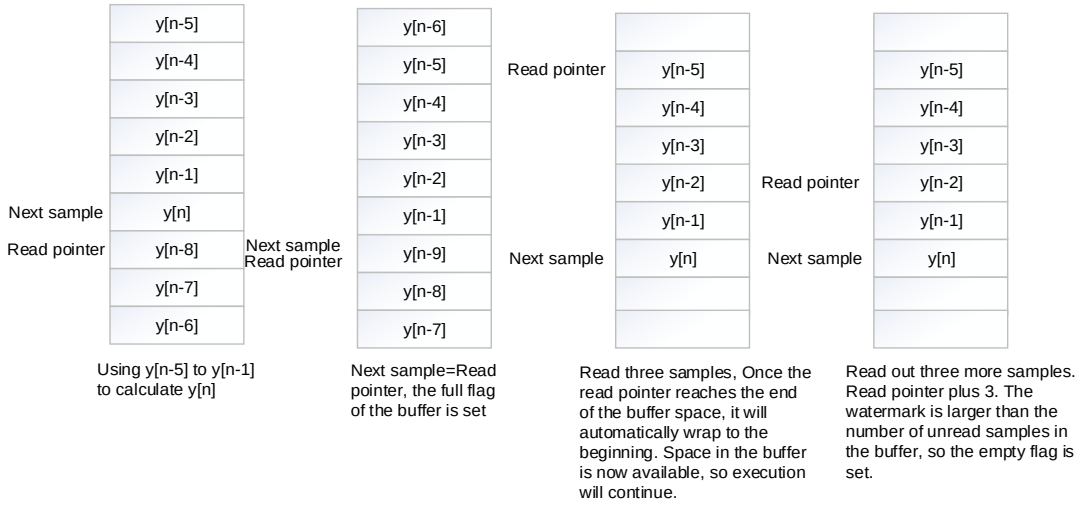
The Y buffer could work in circular buffer mode. The buffer full flag is set if the address for next output data is equal to the address which is indicated by the read pointer, and operation is stalled until the output data is read.

For IIR filters, the next output sample y[n] is calculated by M previous output samples y[n-M] to y[n-1], the least recent sample y[n-M] is discarded when a new sample is added.

If the watermark threshold, which is programmed in the Y_WBFF field of the FAC_YBCFG register, is no less than the number of unread data, the Y buffer is regarded as empty. In case that the empty flag is not set, if interrupt or DMA is enabled, the requests are generated to read data from the Y buffer. Without risk of underflow, several data could be transferred under one interrupt for watermark. However, in case of underflow occurring, UFEF flag is set and the read pointer stops incrementing, while the content of memory, which is addressed at the read pointer, is returned for read operation.

*Figure 47-6 Circular output buffer area* shows the change process of Y buffer during an 5-tap IIR filtering processing, while the watermark is set to 3.

**Figure 47-6 Circular output buffer area**




## 47.3.5. Initialization functions

The FAC unit is initialized by writing the proper value in the FUN bit field of the FAC_PARACFG register when the EXE bit is setting. The IPP and IPQ bitfields contain values that need to be preloaded, while IPR bit field is not used. The EXE bit is reset by hardware automatically, as the Initialization function completed.

DMA requests and interrupts should be disabled during initialization, since flow control is not required, data can be transferred to FAC memory through DMA transfer or software.

**X0 buffer loading function**

This loading function pre-loads values from the address X0_BASE, and the write data is loaded into the X0 buffer from FAC_WDATA register, at the same time, the write address is increasing. When N values have been loaded into the X0 buffer, the write pointer finally points

to the address X0_BASE + N. The parameter IPP contains N, the number of values, which is loaded into the X0 buffer, while IPQ and IPR are not used. This function is completed when the N write operations to the FAC_WDATA register are completed.

**X1 buffer loading function：**

This loading function pre-loads values from the address X1_BASE, and the write data is loaded into the X1 buffer from FAC_WDATA register, at the same time, the write address is increasing. In IIR filter, N feed-forward and M feed-back coefficients are loaded into X1 buffer, The coefficients sum is N+M. In FIR filter, since feedback coefficients are absent, M is equal to 0, N feed-forward coefficients are loaded into X1 buffer.

The parameter IPP contains N feed-forward coefficients and the parameter IPQ contains M feed-back coefficients, Both IPP and IPQ are loaded into X1 buffer, where the starting address of IPP is X1_BASE and the starting address of IPQ is X1_BASE+N. IPR is not used in loading X1 buffer. This function is completed when the N + M write operations to the FAC_WDATA register are completed.

**Y buffer loading function：**

This loading function pre-loads values from the address Y_BASE, and the write data is loaded into the Y buffer from FAC_WDATA register, at the same time, the write address is increasing the write pointer finally points to the address Y_BASE + N. By this function, the feedback storage element of the IIR filter can be preloaded. The parameter IPP contains N, the number of values, which is loaded into the Y buffer, while IPQ and IPR are not used. This function is completed when the N write operations to the FAC_WDATA register are completed.

### 47.3.6. Filter functions

Writing the appropriate value in the FUN bit field of the FAC_PARACFG register can trigger FIR or IIR filter functions when EXE bit is setting. The IPP, IPQ, and IPR fields contain the suitable parameter values for each filter function, The filter function runs all the time except that the software resets the EXE bit.

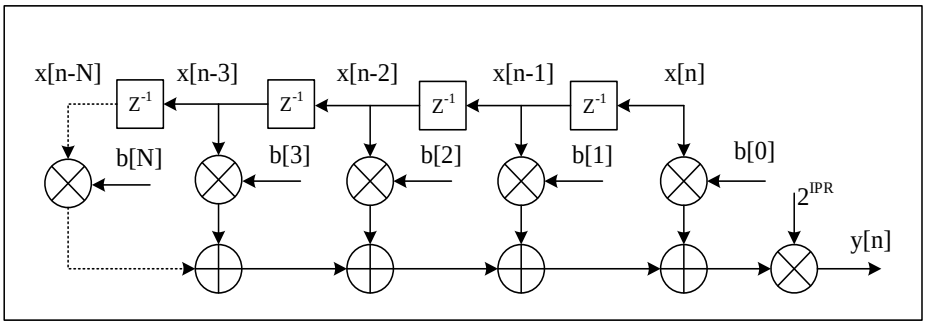
**FIR filter: $\underline{Y} = \underline{B} * \underline{X}$**

$$y_n = 2^{IPR} \sum_{k=0}^{N} (b_k \times x_{n-k}) \quad (47\text{-}1)$$

vector $\underline{B}$ contains N + 1 filter coefficients, vector $\underline{X}$ contains indefinite length input samples, The elements of $\underline{Y}$ is calculated as the dot product, $y_n = \underline{B}.\underline{X}_n$, and $\underline{X}_n = [x_{n-N},...,x_n]$. This function conforms to a finite impulse response (FIR) filter.

FIR filter structure is shown as ***Figure 47-7 The structure of FIR***.

**Figure 47-7 The structure of FIR filter function**




X0 buffer is a circular buffer and composed of the elements of vector X, the length of buffer is N + 1 + d( d is the length of headroom). X1 buffer is composed of the elements of vector B, the length of buffer is N + 1. Y buffer is a circular buffer and composed of the output values($y_n$), the length of buffer is d. The length of the parameter IPP is N+1, The vector B is in the range[2:127]. The parameter IPR is the gain, applied to the accumulator output by multiplied $2^{IPR}$, where IPR is in the range [0:7]. Parameter IPQ is not used.

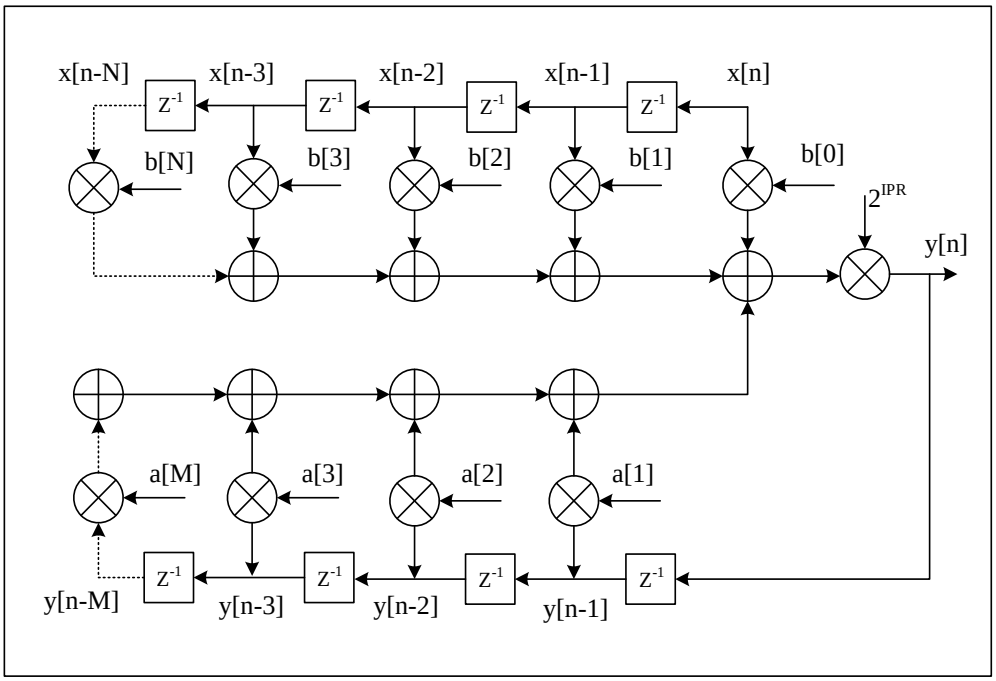
### IIR filter: Y = B*X+A*Y

$$y_n=2^{IPR}\left(\sum_{k=0}^{N}(x_{n-k}\times b_k)+\sum_{k=1}^{M}\left(y_{n-k}\times a_k\right)\right) \quad (47\text{-}2)$$

The infinite impulse response (IIR) filter output vector Y which is the convolution of a coefficient vector B (length is N+1) and a vector X (length is indefinite), add the convolution of vector Y with vector A (length is M). The elements of Y are calculated by B.$X_n$ + A.$Y_{n-1}$, where $X_n$ = [$x_{n-N}$,...,$x_n$] is composed of the N+1 elements, while $Y_{n-1}$= [$y_{n-M}$,...,$y_{n-1}$] is composed of the M elements.

IIR filter structure is shown as ***Figure 47-8 The structure of IIR*** .

**Figure 47-8 The structure of IIR filter**




X0 buffer is a circular buffer and composed of the elements of vector X, the length of buffer is N + 1 + d. X1 buffer is composed of the elements of vector B and A, the length of buffer is M + N + 1. Y buffer is a circular buffer and composed of the output values($y_n$), the length of buffer is M + d. The length of parameter IPP is N+1, the coefficient vector B is in the range [2:64]. The parameter IPR is the gain, applied to the accumulator output by multiplied $2^{IPR}$, where IPR is in the range [0:7]. The length of parameter IPQ is M, the coefficient vector A is in the range [1:63].

## 47.3.7. Fixed point data format

The FAC operates input and output values in fixed signed integer format (q1.15), 1 is integer sign and 15 is fractional. The numeric range is from -1 (0x8000) to 1 - $2^{-15}$ (0x7FFF).

The accumulator value format is q4.22, the format value contains 26 bits, 4 are integer sign and 22 are fractional, support accumulation sums in the range from -8 (0x2000000) to +7.99999976(0x1FFFFFF). In steps of 6dB, a programmable gain can be used to the accumulator output, from 0dB to 42dB.

If the numeric range is exceeded, the accumulator content is not saturated. If value is more than +7.99999976 or less than -8, triggered wrap is harmless, since subsequent accumulations undo the wrapping. However, if wrapping occurs, STEF flag in the FAC_STAT register is set, and then, if STEIE bit in the FAC_CTL register is set, corresponding interrupt is generated.

When the CPEN bit of FAC_CTL register is set, the data output of accumulator could be saturated optionally after the programmable gain application. Any value is forced output as 1 - $2^{-15}$ or -1 while the value exceeds the q1.15 numeric range. While using the gain, If CPEN

bit is not set, the unused accumulator bits are truncated simply.

### 47.3.8. Float point data format

The operation data and calculation result data format is given in ***Table 47-1 IEEE 32-Bit Single Precision Floating-Point Format***. they must meet IEEE 32 bit Single Precision Floating-Point.

**Table 47-1 IEEE 32-Bit Single Precision Floating-Point Format**

| S [31] | E [30:23] | M [22:0] | Value (V) |
|---|---|---|---|
| 0 | 0 | 0 | Zero (V = 0) |
| 1 | 0 | 0 | Negative Zero (V = -0) |
| 0 + ve<br>1 - ve | 0 | non zero | De-normalized (V=$(-1)^{s}$*$2^{(-126)}$* (0.M)) |
| 0 + ve<br>1 - ve | 1 to 254 | 0 to 0x7FFFFF | Normal Range(V=$(-1)^{s}$*$2^{(E-127)}$* (1.M)) |
| 0 | 254 | 0x7FFFFF | Positive Max (V = +Max) |
| 1 | 254 | 0x7FFFFF | Negative Max (V = -Max) |
| 0 | max=255 | 0 | Positive Infinity (V = +Infinity) |
| 1 | max=255 | 0 | Negative Infinity (V = -Infinity) |
| x | max=255 | non zero | Not A Number (V = NaN) |

The treatment of the various IEEE floating-point numerical formats for this FAC is given as below:

**De-Normalized Numbers:** A de-normalized operand (E=0, M!=0) input is treated as zero (E=0, M=0).

**Overflow:** Overflow occurs when an operation generates a value that is too large to represent in the given floating-point format. Under such cases, a positive or negative Infinity value is returned, and STEF flag in the FAC_STAT register is set.

**Not a Number (NaN):** An NaN operand (E=max, M!=0) input is treated as infinity (E=max, M=0).

**Note:** configure FLTEN bit in FAC_CTL register as set, input and output data in 32 bit IEEE data format, not support clip and gain of output.

### 47.3.9. FIR filters

The FAC supports FIR filters, and the number of taps or coefficients is N. FIR filters require a minimum length of local memory of 2N + 1: N input samples ,1 output sample and N coefficients,. the maximum size for N is 127 while the local memory size is 256,. A small amount of additional space is allocated for maximum throughput, d0 is for input sample buffer and d1 is for output sample buffer, so as to guarantee the filter does not stop to wait for a new input sample or reading the output sample. 2N + d0 + d1 is the required local memory. X0_BUF_SIZE is equal to N + d0, X1_BUF_SIZE is equal to N and Y_BUF_SIZE is equal to

d1 (Y_BUF_SIZE could be equal to 1 if no extra space is needed).

Even though the user can arbitrarily allocate the buffer base address, it is necessary to avoid that the X1 buffer overlap with the other buffer, otherwise, the coefficients are overwritten. For example, X1_BASE is equal to 0, X0_BASE is equal to N, and Y_BASE is equal to 2N+d0. However, the X0 and Y buffers may overlap if the memory space is limited, such as X1_BASE is equal to 0, X0_BASE is equal to N, and Y_BASE is equal to N. The output sample would replace the oldest input sample. Since Y_BUF_SIZE = X0_BUF_SIZE = N + d0, the buffers hold in sync still.

**Note**: The X0_WBFF field of X0BCFG register must be programmed not more than $\log_2(d0)$, or else, before writing N input samples, the buffer is flagged as full, then the samples are no longer needed. In the same way, the Y_WBEF field of YBCFG register must be programmed not more than $\log_2(d1)$.

The X1 buffer must preload the filter coefficients. Any number of samples, which is up to N, could be preloaded into the X0 buffer. Since feedback path is not needed for the FIR filter, pre-loading the Y buffer is unnecessary.

The FAC_CTL register should be programmed depend on polling, interrupt and DMA, which is used for writing data to FAC memory and reading data from FAC memory. In polling method, software should confirm that the X0BFF bit is reset or YBEF bit is reset before writing to WDATA or reading from RDATA. In interrupt method, the interrupt request is launched when the X0BFF bit is reset for writing, or the YBEF bit is reset for reading. In DMA method, DMA requests are launched on the DMA write channel or read channel, while the X0BFF bit is reset or the YBEF bit is reset.

Writing the following values in the FAC_PARACFG register, thus the filter is started.

FUN= 8 (FIR filter); IPP = N (number of coefficients); IPQ = “any value”; IPR= Gain; EXE = 1.

If the number of values preloaded in the X0 buffer is less than $N + d - 2^{X0_WBFF}$, the X0BFF flag is in low state. If the WIE bit is set, writing $2^{X0_WBFF}$ samples into X0 buffer through the FAC_WDATA register is triggered by the write interrupt request. When $2^{X0_WBFF}$ values have been written to FAC_WDATA register, The interrupt handler check the X0BFF flag unless the X0BFF bit in FAC_STAT register is set. In the same way, if DWEN bit is set in the FAC_CTL register, constantly generate DMA write channel request, unless the X0BFF bit in FAC_STAT register is set.

When samples (at least N) have been written into the X0 buffer, First output sample is calculated by the filter. When writing $2^{Y_WBEF}$ output samples into the Y buffer, the YBEF bit in the FAC_STAT register is reset, the interrupt is request to read $2^{Y_WBEF}$ samples from the buffer, if the RIE bit is set in the FAC_CTL register. The interrupt handler should repeatedly check the YBEF flag after every $2^{Y_WBEF}$ values have been read from FAC_RDATA register unless the YBEF bit in FAC_STAT register is set. In the same way, if DREN bit is set in the FAC_CTL register, constantly generate DMA read channel request, unless the YBEF bit in FAC_STAT register is set. Resetting the EXE bit could halt the filter operation, or else, the filter will continue to operate.

### 47.3.10. IIR filters

The FAC supports IIR filters with length N (the number of coefficients or feed-forward taps) and M (the number of feedback coefficients, which can be configured from 1 to N-1).

The minimum memory required for an IIR filter is 2N + 2M, which include N feed-forward coefficients and M feed-back coefficients, N input samples and M output samples. In case that M is equal to N-1, the maximum filter length N = 64 can be implemented.

For maximum throughput, there is an additional space d0 for input buffer size, and d1 for output buffer size, should be allowed to configured, so the total memory requirement 2M + 2N + d0 + d1. X0_BUF_SIZE = N + d0, X1_BUF_SIZE = N + M and Y_BUF_SIZE = M + d1. The buffer base address must not overlap, even if it is allocated anywhere, such as: X1_BASE = 0, X0_BASE = N+M and Y_BASE = 2N+M+d0.

**Note**: The WBFF field of X0BCFG register must be programmed not more than $\log_2(d0)$, or else, before writing N input samples, the full flag of the buffer is set, and then no more sample points are needed. In the same way, the WBEF field of YBCFG register must be programmed not more than $\log_2(d1)$.

By using the Load X1 buffer function, the X1 buffer must preload the filter coefficients (N feed-forward and M feedback). Any number of samples, which is up to a maximum of N, could be preloaded into the X0 buffer. In the same way, any number of samples, which is up to a maximum of M, could be preloaded into the Y buffer.

Writing the following values in the FAC_PARACFG register, thus the filter is started.

FUN= 9 (IIR filter); IPP = N (number of feed-forward coefficients);

IPQ = M (number of feed-back coefficients); IPR= Gain; EXE = 1.

If the number of values, which have been preloaded in the X0 buffer, is less than N + d – $2^{X0_WBFF}$, the X0BFF flag is held in low state. If the WIE bit is set in the FAC_CTL register, When $2^{X0_WBFF}$ values have been written to FAC_WDATA register, The interrupt handler check the X0BFF flag unless the X0BFF bit in FAC_STAT register is set. In the same way, if DWEN bit is set in the FAC_CTL register, constantly generate DMA write channel request, unless the X0BFF bit in FAC_STAT register is set.

When samples (at least N) have been written in the X0 buffer, first output sample is calculated by the filter, which is calculated by using the X0 buffer first N samples and the Y buffer first M samples. The address at where first output sample write into Y buffer is Y_BASE+M.

When writing $2^{Y_WBEF}$ output samples in the Y buffer, the YBEF bit in the FAC_STAT register is reset. the interrupt request to read $2^{Y_WBEF}$ samples from the buffer, if the RIE bit is set in the FAC_CTL register. The interrupt handler should repeatedly check the YBEF flag after every $2^{Y_WBEF}$ values have been read from FAC_RDATA register unless the YBEF bit in FAC_STAT register is set. In the same way, if DREN bit is set in the FAC_CTL register, constantly generate DMA read channel request, unless the YBEF bit in FAC_STAT register is set. Resetting the EXE bit could halt the filter operation, or else, the filter will continue to

operate.

## 47.4. Register definition

FAC base address: 0x4802 4800

### 47.4.1. FAC X0 buffer configure register (FAC_X0BCFG)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit) and can only be modified if EXE = 0 in the FAC_PARACFG register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | X0_WBFF[1:0] | | Reserved | | | | | | | |
| | | | | | | rw | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X0B_SIZE[7:0] | | | | | | | | X0B_ADDR[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25:24 | X0_WBFF[1:0] | Buffer full flag of watermark.<br>If free spaces number in the buffer is less than $2^{X0_WBFF}$, the flag is set.<br>00: Threshold = 1(if DMA write requests are enabled)<br>01: Threshold = 2<br>10: Threshold = 4<br>11: Threshold = 8<br>Under one interrupt, if several data would be transferred into the buffer, the threshold should be set more than 1. |
| 23:16 | Reserved | Must be kept at reset value. |
| 15:8 | X0B_SIZE[7:0] | X0 buffer size, the number of feed-forward taps. |
| 7:0 | X0B_ADDR[7:0] | X0 buffer base address |

### 47.4.2. FAC X1 buffer configure register (FAC_X1BCFG)

Address offset: 0x04
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| X1B_SIZE[7:0] | X1B_ADDR[7:0] |
|---|---|
| rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:8 | X1B_SIZE[7:0] | X1 buffer size<br>When FAC is running (EXE = 1), this bit field can not be changed. |
| 7:0 | X1B_ADDR[7:0] | X1 buffer base address<br>When FAC is running (EXE = 1), this bit field can be changed. For example, When changing the coefficient value, the filter should be paused, because changing the factor during the calculation will affect the results. |

### 47.4.3. FAC Y buffer configure register (FAC_YBCFG)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit) and can only be modified if EXE = 0 in the FAC_PARACFG register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | Y_WBEF[1:0] | | Reserved | | | | | | | |
| | | | | | | rw | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| YB_SIZE[7:0] | | | | | | | | YB_ADDR[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:26 | Reserved | Must be kept at reset value. |
| 25:24 | Y_WBEF[1:0] | Buffer empty flag of watermark<br>If the number of unread data in the buffer is less than $2^{Y_WBEF}$, the flag is set.<br>00: Threshold = 1<br>01: Threshold = 2<br>10: Threshold = 4<br>11: Threshold = 8<br>Under one interrupt, if several data would be transferred from the buffer, setting the threshold more than 1.<br>If DMA read command is enabled, threshold should be set to 1. |
| 23:16 | Reserved | Must be kept at reset value. |
| 15:8 | YB_SIZE[7:0] | Y buffer size<br>The minimum buffer size is the watermark threshold + 1 for FIR filters.<br>the minimum buffer size is the watermark threshold + the number of feedback taps |

| | | |
|---|---|---|
| | | for IIR filters. |
| 7:0 | YB_ADDR[7:0] | Y buffer base address |

### 47.4.4. FAC Parameter configure register (FAC_PARACFG)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXE | FUN[6:0] | | | | | | | IPR[7:0] | | | | | | | |
| rw | rw | | | | | | | rw | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IPQ[7:0] | | | | | | | | IPP[7:0] | | | | | | | |
| rw | | | | | | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | EXE | Execution<br>0: Stop execution<br>1: Start execution<br>This bit is set in order to execute the function of the FUN bit field, FAC will stop any ongoing functions through software reset. This bit is reset by hardware for initialization functions. |
| 30:24 | FUN[6:0] | Function<br>0000001: Load X0 buffer<br>0000010: Load X1 buffer<br>0000011: Load Y buffer<br>0001000: FIR filter<br>0001001: IIR filter<br>others: Reserved<br>This register can only be modified when EXE = 0 in the FAC_PARACFG register. |
| 23:16 | IPR[7:0] | Input parameter IPR<br>This register can only be modified when EXE = 0 in the FAC_PARACFG register. |
| 15:8 | IPQ[7:0] | Input parameter IPQ<br>This register can only be modified when EXE = 0 in the FAC_PARACFG register. |
| 7:0 | IPP[7:0] | Input parameter IPP<br>This register can only be modified when EXE = 0 in the FAC_PARACFG register. |

### 47.4.5. FAC Control register (FAC_CTL)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | RST |
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPEN | FLTEN | Reserved | | | | DWEN | DREN | Reserved | | GSTEIE | STEIE | UFEIE | OFEIE | WIE | RIE |
| rw | rw | | | | | rw | rw | | | rw | rw | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:17 | Reserved | Must be kept at reset value. |
| 16 | RST | Reset FAC unit<br>0: Reset disable<br>1: Reset enable<br>The write pointer and read pointer, EXE bit, FAC_STAT register and FAC_PARACFG register will be reset when RST is 1. |
| 15 | CPEN | Clipping enable<br>0: Clipping disable, the value of accumulator output out of range is truncated<br>1: Clipping enable, the value of accumulator output out of range is saturated to the maximum positive value or maximum negative value. |
| 14 | FLTEN | Floating point format enable<br>0: Input data and result support fixed point data format q1.15<br>1: Input data and result support floating point data format<br>This bit can only be modified if EXE = 0 in the FAC_PARACFG register. |
| 13:10 | Reserved | Must be kept at reset value. |
| 9 | DWEN | DMA write channel enable<br>0: DMA request is not generated<br>1: DMA request is generated while the X0 buffer is not full.<br>This bit can only be modified if EXE = 0 in the FAC_PARACFG register. |
| 8 | DREN | DMA read channel enable<br>0: DMA request is not generated<br>1: DMA request is generated while the Y buffer is not empty.<br>This bit can only be modified if EXE = 0 in the FAC_PARACFG register. |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | GSTEIE | Gain saturation error interrupt enable<br>0: No interrupts are generated.<br>1: An interrupt request is generated while the GSTEF flag is set<br>Software set and clear this bit. |
| 4 | STEIE | Saturation error interrupt enable<br>0: No interrupts are generated.<br>1: An interrupt request is generated while the STEF flag is set |

| | | |
|---|---|---|
| | | Software set and clear this bit. |
| 3 | UFEIE | Underflow error interrupt enable<br>0: No interrupts are generated.<br>1: An interrupt request is generated while the UFEF flag is set<br>Software set and clear this bit. |
| 2 | OFEIE | Overflow error interrupt enable<br>0: No interrupts are generated.<br>1: An interrupt request is generated while the OFEF flag is set<br>Software set and clear this bit. |
| 1 | WIE | Write interrupt enable<br>0: No interrupts are generated.<br>1: An interrupt request is generated if the X0BFF flag is set<br>Software set and clear this bit. |
| 0 | RIE | Read interrupt enable<br>0: No interrupts are generated.<br>1: An interrupt request is generated if the YBEF flag is set<br>Software set and clear this bit. |

### 47.4.6. FAC Status register (FAC_STAT)

Address offset: 0x14

Reset value: 0x0000 0001

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | GSTEF | STEF | UFEF | OFEF | Reserved | | | | | | X0BFF | YBEF |
| | | | | | r | r | r | | | | | | | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11 | GSTEF | Gain saturation error flag, it is set when gain exceed range<br>0: No gain saturation error detected<br>1: Gain saturation error detected. |
| 10 | STEF | Saturation error flag<br>0: No saturation error detected<br>1: Saturation error detected.<br>Saturation occurs when the cumulative result exceeds the range of the value |
| 9 | UFEF | Underflow error flag |

| | | |
|---|---|---|
| | | 0: No underflow error detected<br>1: Underflow error detected.<br>When there is no valid data available in the Y buffer, underflow occurs when reading from FAC_RDATA |
| 8 | OFEF | Overflow error flag<br>0: No overflow error detected<br>1: Overflow error detected<br>When there is no free space in X1 buffer, overflow occurs when writing to FAC_WDATA. |
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | X0BFF | X0 buffer full flag<br>0: X0 buffer is not full.<br>1: X0 buffer is full.<br>Hardware or a reset will set and clear this bit. |
| 0 | YBEF | Y buffer empty flag<br>0: Y buffer is not full.<br>1: Y buffer is full.<br>Hardware or a reset will set and clear this bit. |

### 47.4.7. FAC write data register (FAC_WDATA)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| WDATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

When FLTEN = 1, float point data format is selected;

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | WDATA[31:0] | Write data<br>When a write command is performed on the register, the write data is transferred to the address offset which is pointed to by the write pointer. After each write of data is completed, The pointer address is incremented. |

When FLTEN = 0, fix point data format is selected;

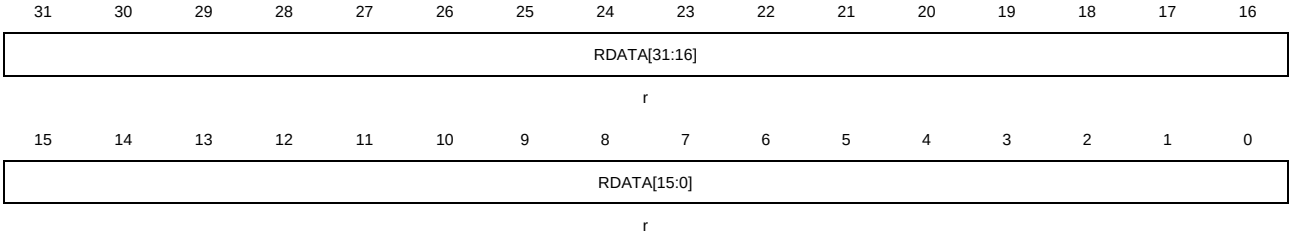
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | WDATA[15:0] | Write data |

When a write command is performed on the register, the write data is transferred to the address offset which is pointed to by the write pointer. After each write of data is completed, The pointer address is incremented.

### 47.4.8. FAC read data register (FAC_RDATA)

Address offset: 0x1C
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




When FLTEN = 1, float point data format is selected;

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | RDATA[31:0] | Read data<br>When a read command is performed on the register, the contents in the Y buffer which is pointed to by the read pointer are the read data. When each read data is completed, The pointer address is incremented. |

When FLTEN = 0, fix point data format is selected;

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | RDATA[15:0] | Read data<br>When a read command is performed on the register, the contents in the Y buffer which is pointed to by the read pointer are the read data. When each read data is completed, The pointer address is incremented. |

# 48. Hardware semaphore (HWSEM) (For GD32H737_757_759)

## 48.1. Overview

Hardware semaphore (HWSEM) provides a non-blocking mechanism to ensure the synchronization of processes. HWSEM realizes 32 semaphores in an atomic way, supporting semaphore write lock and read lock, and semaphore can only be unlocked when bus master and process are matched.

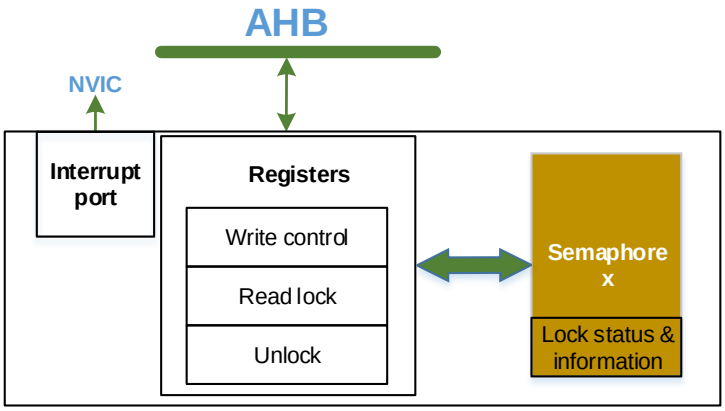
## 48.2. Characteristics

- 32 semaphores.
- An interrupt is generated when a semaphore is unlocked.
- Semaphore is unlocked only when MID[3:0] and PID[7:0] are matched.

## 48.3. Function overview

### 48.3.1. Block diagram

**Figure 48-1. HWSEM diagram**




As shown in ***Figure 48-1. HWSEM diagram***, HWSEM includes three sub-blocks:

- Semaphore x sub-block, including semaphore lock status and lock informations.
- Registers, used for accessing the semaphores.
- Interrupt port for interrupt control and status.

### 48.3.2. Semaphore x

#### Lock status

LK bit in HWSEM_CTLx (x = 0…31) register shows the lock status of semaphore x.

When LK bit is 0, the semaphore is unlocked.

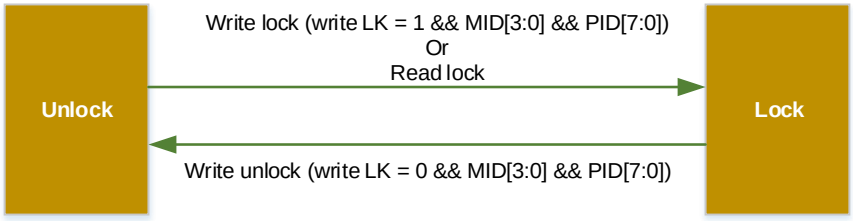
When LK bit is 1, the semaphore is locked.

#### Lock information

When the semaphore is unlocked, MID[3:0] and PID[7:0] bit fields in HWSEM_CTLx (x = 0…31) register are 0.

When the semaphore is locked, MID[3:0] bit field in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register shows which AHB bus master locked it, and PID[7:0] bit field in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register shows which process of that AHB bus master locked it.

**Figure 48-2. HWSEM state machine**




### 48.3.3. Lock a semaphore

HWSEM supports two ways for semaphore locking:

- Write lock.
- Read lock.

**Note:** Only MID[3:0] which is equal to 0b1011 can lock and unlock a semaphore.

#### Write lock

When a semaphore is locked by write lock, MID[3:0] bit field in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register is the AHB bus master which locked it, and PID[7:0] bit field in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register is the configured process identification.

Each process of AHB bus master must owns an unique PID[7:0] value.

Write lock operation follows the following steps:

1. Write LK = 1, and write MID[3:0] and PID[7:0] bit fields in HWSEM_CTLx (x = 0…31)

register to lock the semaphore.

2. Read the written HWSEM_CTLx (x = 0…31) register, to confirm the MID[3:0] and PID[7:0] bit fields are the written value or not.
3. If the MID[3:0] and PID[7:0] bit fields in HWSEM_CTLx (x = 0…31) register match with the written value, the semaphore is locked by this process. Otherwise, the semaphore is locked by other AHB bus master or process, go to step 1 to try again.

#### Read lock

When a semaphore is locked by read lock, MID[3:0] bit field in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register is the AHB bus master which locked it, while PID[7:0] bit field in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register is always 0.

If more than one processes use read lock to lock the same semaphore, when the semaphore is read locked by one process, other processes will find that the semaphore is read as locked by themselves, and cannot recognize which process locks it.

Read lock operation follows the following steps:

1. Read HWSEM_RLKx (x = 0…31) register to lock the semaphore.
2. If the MID[3:0] matches with the AHB bus master, and PID[7:0] is read as 0, the semaphore is read locked. While if the semaphore is locked by other AHB bus master (the MID[3:0] does not match with the AHB bus master) or write locked by other processes (PID[7:0] is not 0), go to step 1 to try again.

### 48.3.4. Unlock a semaphore

When a semaphore is locked, use write unlock to release a semaphore. The operation is protected from being unlocked by the process which does not have the unlocking priority.

Only the process which has the matching MID[3:0] and PID[7:0] can unlock the semaphore. After a valid write unlock operation, LK bit, MID[3:0] bits and PID[7:0] bits in HWSEM_CTLx (x = 0…31) / HWSEM_RLKx (x = 0…31) register are cleared to 0, an interrupt will be generated if SIEx bit in HWSEM_INTEN register is set to 1.

Write unlock operation follows the following steps:

1. Write LK = 0, and write MID[3:0] and PID[7:0] bit fields in HWSEM_CTLx (x = 0…31) register to unlock the semaphore.
2. Read the written HWSEM_CTLx (x = 0…31) register, to confirm the LK bit is 0.
3. If the MID[3:0] and PID[7:0] bit fields in HWSEM_CTLx (x = 0…31) register match with the semaphore lock information, LK bit is 0 and the semaphore is unlocked. Otherwise, the unlock operation will be ignored and LK bit remains 1.

### 48.3.5. Unlock all semaphores

When the AHB bus master works incorrectly, all unlock operation can be used to clear lock status and lock information of all semaphores.

All unlock operation follows the following steps:

1. Write KEY[15:0] and MID[3:0] bit field of HWSEM_UNLK register to unlock all semaphores.
2. Read the HWSEM_CTLx (x = 0…31) registers, to confirm all the LK bits are 0.
3. If the MID[3:0] bit field in HWSEM_CTLx (x = 0…31) register matches with the semaphore lock information, LK bit is 0 and the semaphore is unlocked, an interrupt will be generated if SIEx bit in HWSEM_INTEN register is set to 1. Otherwise, the unlock operation will be ignored and LK bit remains 1.

### 48.3.6. Interrupts

The semaphore should be locked and unlocked in user process, it cannot be locked or unlocked in the interrupt service. It is suggested to try to lock a semaphore twice, and only when failed on the second try, enable the semaphore interrupt and wait the interrupt.

***Figure 48-3. First try lock success*** shows when target process first try to lock a semaphore, and succeed to lock it, then the clear flag status operation should be done, and no need to enable the semaphore interrupt.

***Figure 48-4. Second try lock success*** shows when target process fails on the first try of locking a semaphore, then before the second try, the clear flag status operation should be done, then target process succeed to lock it, in this situation, there is also no need to enable the semaphore interrupt.

***Figure 48-5. Second try lock fail*** shows when target process fails on the first try of locking a semaphore, then before the second try, the clear flag status operation should be done, however, target process failed to lock it, in this situation, enable the semaphore interrupt and wait the interrupt, then perform the next try.

**Figure 48-3. First try lock success**

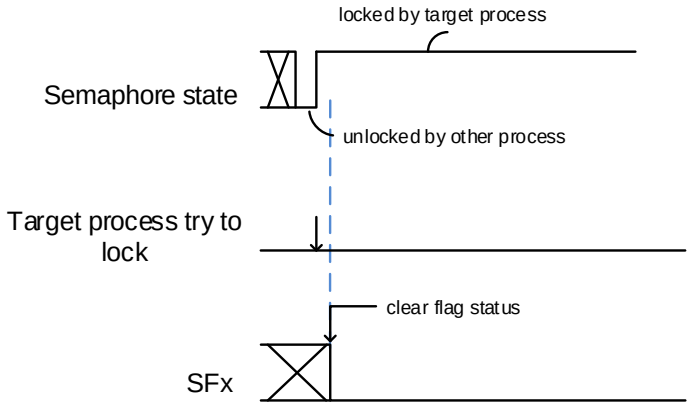


**Figure 48-4. Second try lock success**

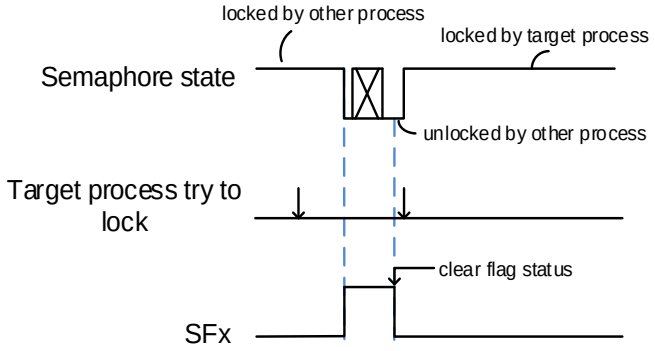
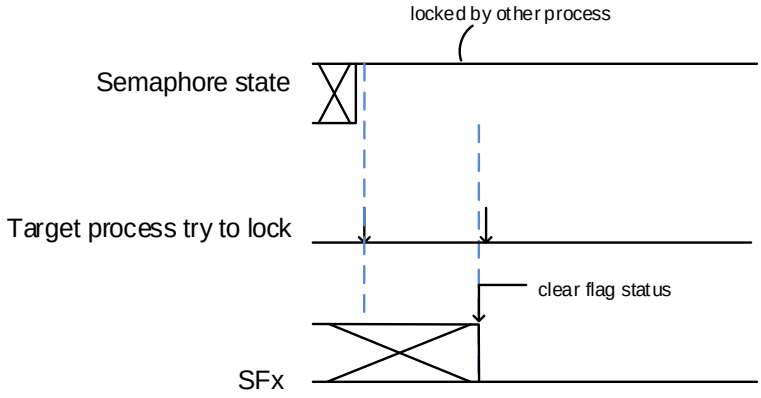



**Figure 48-5. Second try lock fail**



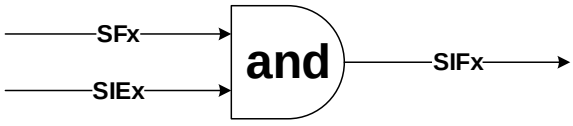

Each semaphore can generate an unlock semaphore interrupt. Each interrupt event has a interrupt flag bit is in HWSEM_INTF register, a clear bit in HWSEM_INTC register, and an enable bit in HWSEM_INTEN register. The relationship is described in the following ***Table 48-1. interrupt events***.

**Table 48-1. interrupt events**

| Interrupt event | Interrupt flag bit | Clear bit | Enable bit |
|---|---|---|---|
| | HWSEM_INTF | HWSEM _INTC | HWSEM _INTEN |
| Semaphore x unlocked | SIFx | SIFCx | SIEx |

The HWSEM interrupt logic is shown in the ***Figure 48-6. HWSEM interrupt logic***, an interrupt can be produced when an interrupt event occurs and enabled the event.

**Figure 48-6. HWSEM interrupt logic**




**Note:** "x" indicates semaphore number (x=0…31).

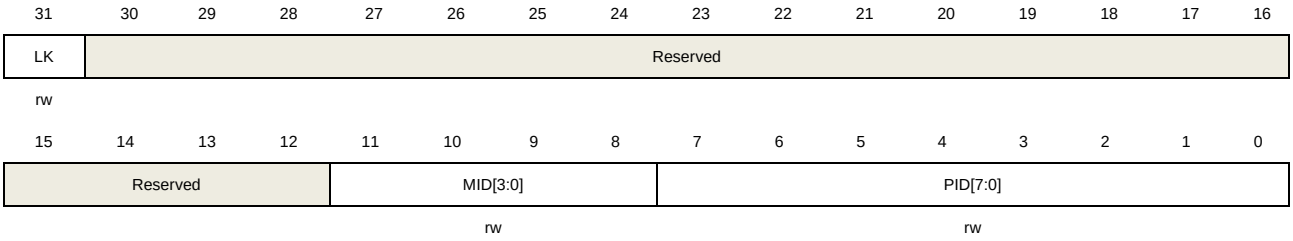
## 48.4. Register definition

HWSEM base address: 0x5802 6400

### 48.4.1. Control register (HWSEM_CTLx)(x = 0…31)

Address offset: 0x00 + 0x4 * x
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



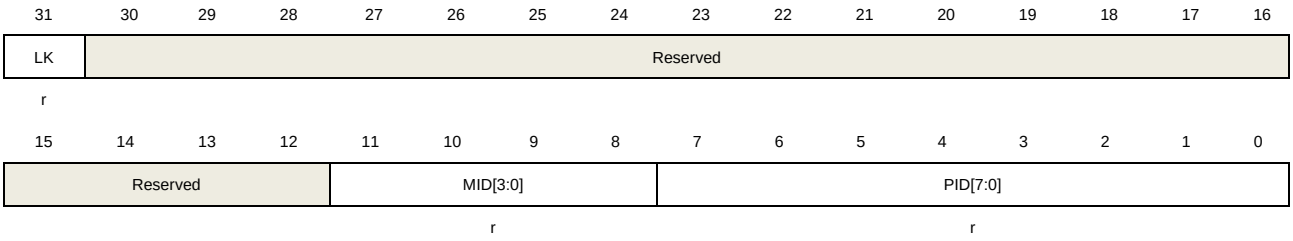

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | Lock semaphore<br>0: Unlock the semaphore when MID[3:0] and PID[7:0] are matched with the lock information<br>1: Try to lock the semaphore |
| 30:12 | Reserved | Must be kept at reset value. |
| 11:8 | MID[3:0] | AHB bus master identification<br>Write by software, only when the semaphore is unlocked, and the writing value of MID[3:0] is matched with the AHB bus master in a write lock operation, the value will be written to this bit field.<br>When the writing value of MID[3:0] is matched with the AHB bus master in a write unlock operation, this bit field will be cleared to 0. |
| 7:0 | PID[7:0] | Process identification<br>Write by software, only when the semaphore is unlocked, the writing value of PID[7:0] will be written to this bit field in a write lock operation.<br>In a write unlock operation, this bit field will be cleared to 0. |

### 48.4.2. Read lock register (HWSEM_RLKx)(x = 0...31)

Address offset: 0x80 + 0x4 * x
Reset value: 0x0000 0000

HWSEM_RLKx (x = 0…31) register access the same physical addresses as the HWSEM_CTLx (x = 0…31) register.

This register has to be accessed by word (32-bit).



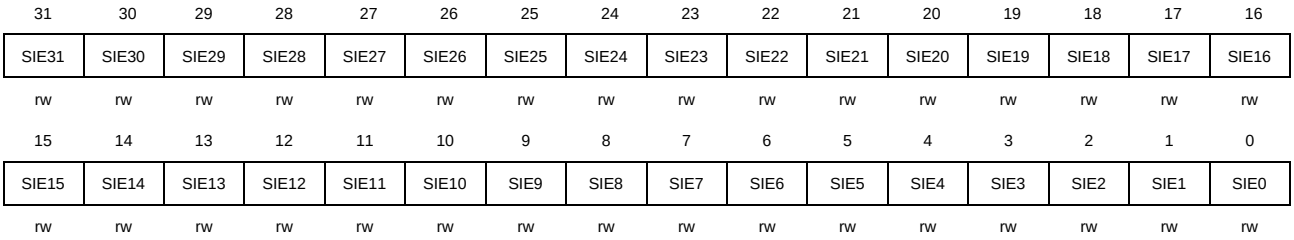

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | LK | Lock semaphore by reading<br>When read this bit by a matched AHB bus master, it always returns 1, if the semaphore is unlocked before read, the semaphore will be read locked by hardware, while if the semaphore is already locked before read, this bit will not change.<br>0: Semaphore is unlocked.<br>1: Semaphore is locked. |
| 30:12 | Reserved | Must be kept at reset value. |
| 11:8 | MID[3:0] | Bus master identification<br>When read this bit field by a matched AHB bus master, if the semaphore is unlocked before read, this bit field will be written with the AHB bus master identification, if the semaphore is locked before read, the MID[3:0] of which locks the semaphore will be returned. |
| 7:0 | PID[7:0] | Process identification<br>When read this bit field by a matched AHB bus master, if the semaphore is unlocked before read, this bit field will be written with 0 and also be read as 0, if the semaphore is locked before read, the PID[7:0] of which locks the semaphore will be returned. |

### 48.4.3. Interrupt enable register (HWSEM_INTEN)

Address offset: 0x100

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SIEx | Semaphore interrupt enable bit<br>0: Disable semaphore interrupt |

1: Enable semaphore interrupt

### 48.4.4. Interrupt flag clear register (HWSEM_INTC)

Address offset: 0x104

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIFC31 | SIFC30 | SIFC29 | SIFC28 | SIFC27 | SIFC26 | SIFC25 | SIFC24 | SIFC23 | SIFC22 | SIFC21 | SIFC20 | SIFC19 | SIFC18 | SIFC17 | SIFC16 |
| w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIFC15 | SIFC14 | SIFC13 | SIFC12 | SIFC11 | SIFC10 | SIFC9 | SIFC8 | SIFC7 | SIFC6 | SIFC5 | SIFC4 | SIFC3 | SIFC2 | SIFC1 | SIFC0 |
| w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 | w1 |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SIFCx | Semaphore interrupt flag clear bit<br>Written by software, and be read as 0.<br>0: No effect<br>1: Clear the semaphore flag and interrupt flag |

### 48.4.5. Status register (HWSEM_STAT)

Address offset: 0x108

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SF31 | SF30 | SF29 | SF28 | SF27 | SF26 | SF25 | SF24 | SF23 | SF22 | SF21 | SF20 | SF19 | SF18 | SF17 | SF16 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SF15 | SF14 | SF13 | SF12 | SF11 | SF10 | SF9 | SF8 | SF7 | SF6 | SF5 | SF4 | SF3 | SF2 | SF1 | SF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SFx | Semaphore flag<br>Set by hardware, and clear by writing 1 to SIFCx bit in HWSEM_INTC register.<br>0: No semaphore unlock event occurs<br>1: A semaphore unlock event occurs |

### 48.4.6. Interrupt flag register (HWSEM_INTF)

Address offset: 0x10C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIF31 | SIF30 | SIF29 | SIF28 | SIF27 | SIF26 | SIF25 | SIF24 | SIF23 | SIF22 | SIF21 | SIF20 | SIF19 | SIF18 | SIF17 | SIF16 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIF15 | SIF14 | SIF13 | SIF12 | SIF11 | SIF10 | SIF9 | SIF8 | SIF7 | SIF6 | SIF5 | SIF4 | SIF3 | SIF2 | SIF1 | SIF0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SIFx | Semaphore interrupt flag status<br>Set by hardware, and clear by writing 1 to SIFCx bit in HWSEM_INTC register.<br>0: No pending interrupt<br>1: An interrupt is pending |

## 48.4.7. Unlock register (HWSEM_UNLK)

Address offset: 0x140

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| w | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | MID[3:0] | | | | Reserved | | | | | | | |
| | | | | w | | | | | | | | | | | |

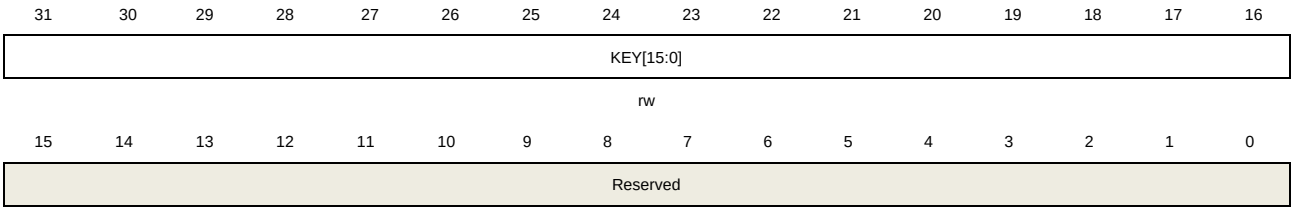
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | KEY[15:0] | Key value<br>Write only, and read as 0.<br>Only when the value of KEY[15:0] is matched with the KEY[15:0] bits in HWSEM_KEY register, all semaphores of the MID[3:0] which is controlled in this register will be unlocked. |
| 15:12 | Reserved | Must be kept at reset value. |
| 11:8 | MID[3:0] | Bus master identification to clear<br>Write only, and read as 0.<br>The semaphores of the AHB bus master which has the written value identiication is to be cleared. |
| 7:0 | Reserved | Must be kept at reset value. |

## 48.4.8. Key register (HWSEM_KEY)

Address offset: 0x144

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | KEY[15:0] | Key for unlocking all semaphores of a bus master<br>The key for unlocking all semaphores. |
| 15:0 | Reserved | Must be kept at reset value. |

# 49. Universal serial bus High-Speed interface (USBHS)

## 49.1. Overview

USB High-Speed (USBHS) controller provides a USB-connection solution for portable devices. USBHS supports both host and device modes, as well as OTG mode with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol). USBHS contains an embedded USB PHY internal which can be configured as High-Speed or Full-Speed. For the GD32H75E, USBHS also provides a ULPI interface for an external USB physical layer (PHY).USBHS supports all the four types of transfer (control, bulk, Interrupt and isochronous) defined in USB 2.0 protocol. There is also a DMA engine operating as an AHB bus master in USBHS to speed up the data transfer between USBHS and system. For Full-Speed operation, battery charging detection (BCD), attach detection protocol (ADP), and link power management (LPM) are also supported.

## 49.2. Characteristics

- Supports USB 2.0 Host mode at High-Speed(480Mb/s), Full-Speed(12Mb/s) or Low-Speed(1.5Mb/s)
- Supports USB 2.0 device mode at High-Speed(480Mb/s) or Full-Speed(12Mb/s)
- Supports OTG protocol with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol)
- Supports all the 4 types of transfer: control, bulk, Interrupt and isochronous
- Supports high-bandwidth interrupt and isochronous transfers
- Includes USB transaction scheduler in host mode to handle USB transaction request efficiently
- Includes a 4KB FIFO RAM
- Supports 16 channels in host mode
- Contains 2 transmit FIFOs (periodic and non-periodic) and one receive FIFO (shared by all channels) in host mode
- Contains 8 transmit FIFOs (one for each IN endpoint) and one receive FIFO (shared by all OUT endpoints) in device mode
- Supports PING protocol in host mode when operates at High-Speed
- Supports 8 OUT and 8 IN Endpoints in device mode
- Supports remote-wakeup in device mode
- Include a USB PHY with OTG protocol supported
- Include an internal DMA scheduler and engine to perform data copy between USBHS and system per the application’s request
- Time intervals of SOFs is dynamic adjusted in host mode
- Able to output SOF pulse to PAD
- Able to detect ID level and VBUS voltage
- Needs external component to supply power for connected USB device in Host mode or

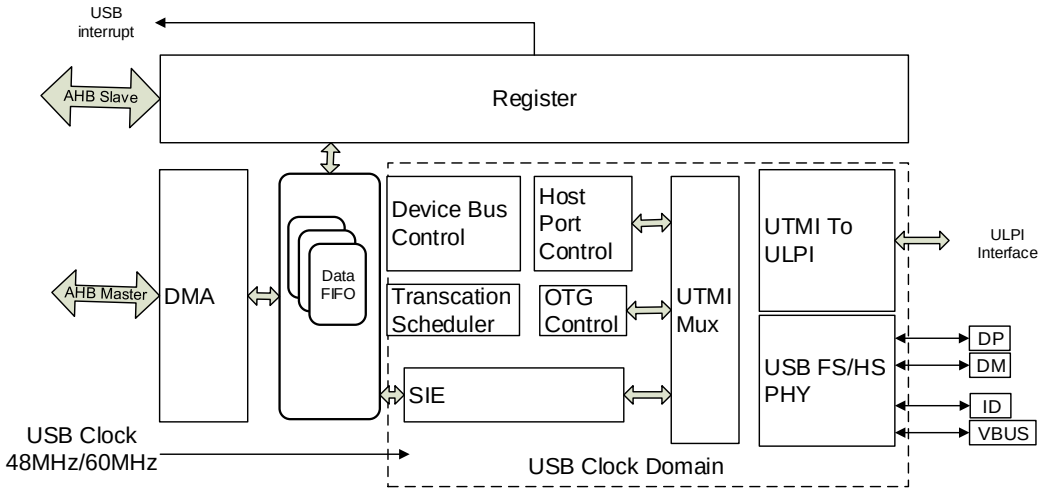
OTG A-device

- Supports charging port detection (BCD) described in Battery Charging Specification Revision 1.2
- Supports Attach Detection Protocol (ADP) described in USB On-The-Go Supplement, Revision 2.0
- Supports Link Power Management (LPM) described in USB 2.0 Link Power Management Addendum and Errata for USB 2.0 ECN: Link Power Management (LPM)

**Note**: For GD32H75E, it supports both the on-chip PHY and an external high-speed PHY via the ULPI interface.

## 49.3. Block diagram

**Figure 49-1. USBHS block diagram**




This series has two USB HS modules(USBHS0 and USBHS1), And both of which support ULPI interface. And high-speed operation is allowed with external HS transceivers. For the GD32H75E, only USBHS0 supports the ULPI interface.

## 49.4. Signal description

**Table 49-1. USBHS signal description**

| I/0 port | Type | Description | Note |
|---|---|---|---|
| VBUS | Input | Bus power port | For internal PHY only |
| DM | Input/Output | Differential data line - port | For internal PHY only |
| DP | Input/Output | Differential data line + port | For internal PHY only |
| ID | Input | USB identification: Mini connector identification port | For internal PHY only |
| ULPI_D[7:0] | Input/Output | ULPI Data line | For external ULPI PHY |
| ULPI_NXT | Input | ULPI next line | For external ULPI PHY |

| I/0 port | Type | Description | Note |
|---|---|---|---|
| ULPI_DIR | Input | ULPI Direction | For external ULPI PHY |
| ULPI_STP | Output | ULPI Stop | For external ULPI PHY |
| ULPI_CLK | Input | ULPI Clock | For external ULPI PHY |

## 49.5. Function overview

### 49.5.1. USBHS PHY selection, clocks and working modes

USBHS can operate as a host, a device or a DRD (Dual-role-Device) and supports two types of connection: internal embedded PHY and external ULPI PHY. The application choose to use either the internal embedded PHY or the external ULPI PHY according to the demand.

The application may limit the maximum speed of the internal PHY or the external ULPI PHY to Full-Speed using SPDFSLS bit in USBHS_HCTL register in host mode or DS[1:0] in USBHS_DCFG register in device mode.

**Table 49-2. USBHS supported speeds**

| Register configuration | | Host supported speed | Device support speed |
|---|---|---|---|
| EMBPHY_FS=1<br>EMBPHY_HS=0<br>(Internal FS PHY) | | Full-Speed<br>Low-Speed | Full-Speed |
| EMBPHY_FS=0<br>EMBPHY_HS=1<br>(Internal HS PHY) | DS =01 (device mode)<br>SPDFSLS=1(host mode) | Full-Speed<br>Low-Speed | Full-Speed |
| | DS =00(device mode)<br>SPDFSLS=0(host mode) | High-Speed<br>Full-Speed<br>Low-Speed | High-Speed<br>Full-Speed |
| EMBPHY_FS=0<br>EMBPHY_HS=0<br>(External ULPI PHY) | DS =01 (device mode)<br>SPDFSLS=1(host mode) | Full-Speed<br>Low-Speed | Full-Speed |
| | DS =00(device mode)<br>SPDFSLS=0(host mode) | High-Speed<br>Full-Speed<br>Low-Speed | High-Speed<br>Full-Speed |

The application control the working modes of USBHS: force host, force device by setting FHM and FDM bits in USBHS_GUSBCS register. When both bits are cleared, USBHS works in OTG mode, which is the default mode after system reset.

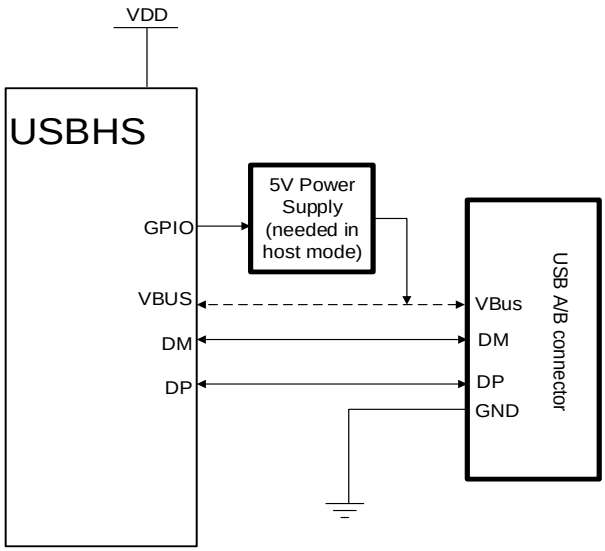
**Internal embedded PHY**

USBHS includes an internal embedded PHY. The internal embedded PHY supports High-Speed, Full-Speed and Low-Speed in host mode, and High-Speed, Full-Speed in device mode, supports OTG protocol with HNP and SRP. Software needs to set EMBPHY_FS bit and reset EMBPHY_HS in USBHS_GUSBCS register to use this PHY in FS mode, or reset EMBPHY_FS bit and set EMBPHY_HS to use this PHY in HS mode. If internal embeddedPHY is selected, the USB clock used for the USBHS needs to be 48MHz in FS

mode and 60MHz in HS mode. The 48MHz USB clock is generated from internal clocks in system, and its source and divider factors are configurable in RCU, while the 60MHz USB clocks is generated from 480MHz PLLUSB.

The pull-up and pull-down resistors are already integrated into the internal PHY and controlled by USBHS automatically based on the current mode (host, device or OTG mode) and connection status. A typical connection using internal PHY is shown in ***Figure 49-2. Connection using internal embedded PHY with host or device mode***.

**Figure 49-2. Connection using internal embedded PHY with host or device mode**



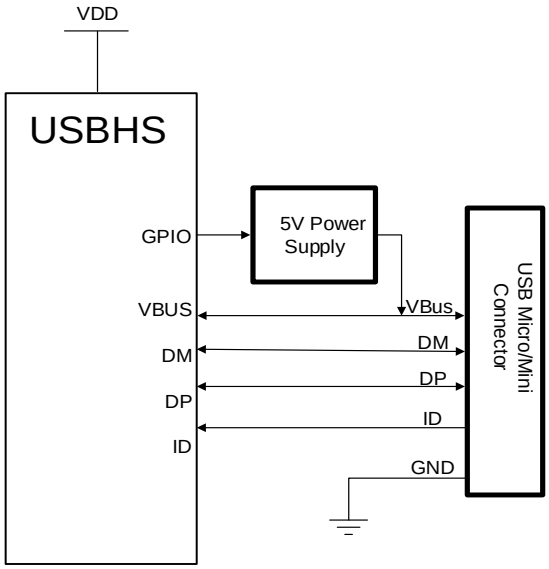

When USBHS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power pin defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuit on VBUS line. If application needs to supply USB power, an external power supply IC is needed. The VBUS connection between USBHS and the USB connector can be omitted in host mode because USBHS doesn't detect the voltage level on VBUS pin and always assumes that the 5V power is present.

When USBHS works in device mode (FHM bit is cleared and FDM bit is set), the VBUS detection circuit is decided by VDEN bit in USBHS_GCCFG register. If the device does not need to detect the voltage on VBUS pin, it may set the VDEN bit and free the VBUS pin for other use. Otherwise, the VBUS connection cannot be omitted, and USBHS continuously monitor the VBUS voltage and will immediately switch off the pull-up resistor on DP line once the VBUS voltage falls below the needed valid value. This will cause a disconnection.

The OTG mode connection is described in the ***Figure 49-3. Connection using internal embedded PHY with OTG mode***. When USBHS works in OTG mode, the FHM, FDM bits in USBHS_GUSBCS and VDEN bit in USBHS_GCCFG should be cleared. In this mode, the USBHS needs all the four pins: DM, DP, VBUS and ID, and uses several voltage comparers to monitor the voltage on these pins. USBHS also includes VBUS charge and discharge circuit to perform SRP request described in OTG protocol. The OTG A-Device or B-Device is decided by the level of ID pins. USBHS controls the pull-up or pull-down resistor during the HNP

protocol.

**Figure 49-3. Connection using internal embedded PHY with OTG mode**

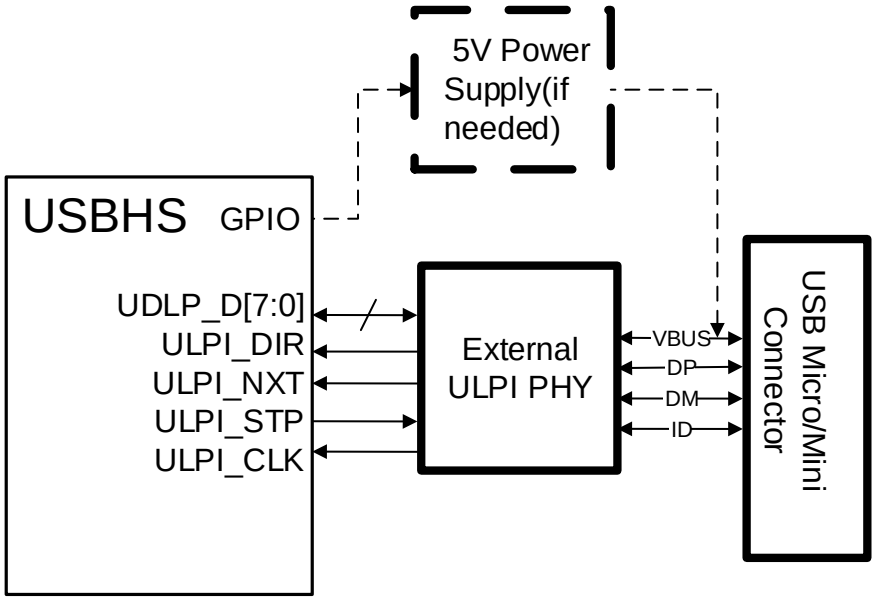



**External ULPI PHY**

USBHS provides a ULPI interface for external PHY integration. An external High-Speed ULPI PHY is needed to support High-Speed USB applications. With external ULPI PHY, USBHS supports High-Speed host and device, all the modes described in internal embedded PHY.

Software needs to clear the EMBPHY_FS bit and EMBPHY_HS bit in USBHS_GUSBCS register to enable the ULPI interface. When ULPI mode enabled, the USB clock which introduced from the ULPI_CLK pin needs to be 60MHz.Software can switch on or off the 60MHz ULPI clock in RCU.

**Figure 49-4. Connection using external ULPI PHY**

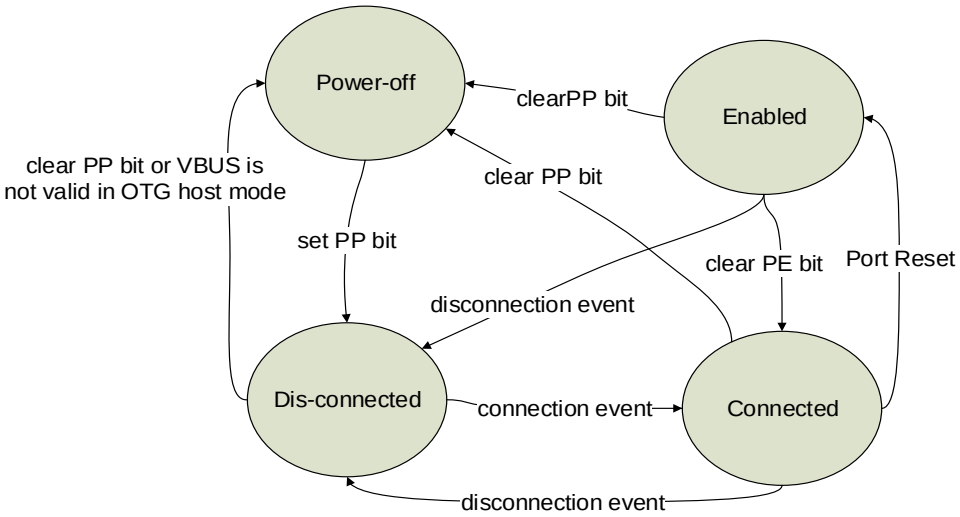


## 49.5.2. USB host function

### USB Host Port State

Host application may control state of the USB port via USBHS_HPCS register. After system initialization the USB port, keep it at power-off state. After PP bit is set by software, the USB PHY (either internal or external) is powered on, and the USB port changes into disconnected state. After a connection is detected, USB port changes into connected state. The USB port changes into enabled state after a port reset is performed on USB bus.

**Figure 49-5. State transition diagram of host port**




### Connection, Reset and Speed identification

As a USB host, USBHS will trigger a connection flag for application after a connection is detected and will trigger disconnection flag after a disconnection event.

PRST bit is used for USB reset sequence. Application may set this bit to start a USB reset and clear this bit to finish the USB reset. This bit only takes effect when port is at connection or enabled state.

The USBHS performs speed identification during connection and reset, and the speed information is reported in PS[1:0] bits in USBHS_HPCS register.

If the maximum supported speed is configured to Full-Speed (SPDFSLS = 1), USBHS only performs speed-identification during device connection process and it identifies the device speed from the voltage level of DM or DP. As is described in USB protocol, Full-Speed device pulls up DP line while Low-Speed device pulls up DM line.

If the maximum supported speed is configured to High-Speed (SPDFSLS = 0), USBHS first performs speed-identification during connection. If a Full-Speed connection is detected, the USBHS will try to perform High-Speed identification (CHIRP sequence described in USB 2.0 protocol) during each USB reset sequence after the connection event. So the application on host should perform a USB reset after a connection event and check the PS[1:0] bits again if it desires to support High-Speed device.

**Suspend and resume**

USBHS supports suspend state and resume operation. When USBHS port is at enabled state, writing 1 to PSP bit in USBHS_HPCS register will cause USBHS to enter suspend state. In suspend state, USBHS stops sending SOFs on USB bus and this will cause the connected USB device to enter suspend state after 3ms. Application can set the PREM bit in USBHS_HPCS register to start a resume sequence to wake up the suspended device and clear this bit to stop the resume sequence. The WKUPIF bit in USBHS_GINTF and the USBHS wake up interrupt will be triggered if a host in suspend state detects a remote wakeup signal.

**SOF generate**

USBHS sends SOF tokens on USB bus in host mode. As described in USB 2.0 protocol, SOF packets are generated (by the host controller or hub transaction translator) every 1ms for Full-Speed links, and every 125 μs for High-Speed links.

Each time after USBHS enters into enabled state, it will send SOF packet using the time defined by USB 2.0 protocol. While, application may adjust the length of a frame or a micro-frame by writing to FRI[15:0] in USBHS_HFT registers. The FRI bits define the number of USB clock cycles in a frame or micro-frame and application should calculate the value based on the frequency of USB clock used by USBHS. The FRT[14:0] bits reflect the remaining clock cycles of the current frame or micro-frame and stops to change during suspend state.

USBHS is able to generate a pulse signal each SOF packet and output it to a pin. The pulse length is 12 HCLK cycle. If application desires to use this function, it needs to set SOFOEN bit in USBHS_GCCFG register and configure the related pin registers in GPIO.

**USB Channels and Transactions**

USBHS includes 16 independent channels in host mode. Each channel is able to communicate with an endpoint in USB device. The transfer type, direction, packet length and other information is configured in channel related registers such as USBHS_HCHxCTL and USBHS_HCHxLEN.

USBHS supports all the four kinds of transfer types: control, bulk, interrupt and isochronous. USB 2.0 protocol divides these transfers into 2 types: non-periodic transfer (control and bulk) and periodic transfer (interrupt and isochronous). Based on this, USBHS includes two request queues: periodic request queue and non-periodic request queue, in order to perform efficient transaction schedule. A request entry in a request queue described above may represent a USB transaction request or a channel operation request.

In non-DMA mode, application needs to write packet into data FIFO via AHB register interface if it wants to start an OUT transaction on USB bus. USBHS hardware will automatically generate a transaction request entry in request queue after the application writes a whole packet. In DMA mode, application only needs to configure the channel property and channel data buffer address, and the DMA engine in USBHS performs the packet data copy and request entry generation. USBHS automatically generate IN request entries when the application enable an IN channel.

The request entries in request queue are processed in order by transaction control module. USBHS always try to process periodic request queue first, then process non-periodic request queue.

After a start of frame USBHS begins to process periodic queue until the queue is empty or bus time required by the current periodic request is not enough, and then process the non-periodic queue. This strategy ensures the bandwidth of periodic transactions in a frame or micro-frame. Each time the USBHS reads and pop a request entry from request queue. If this is a channel disable request, it immediately disables the channel and prepare to process next entry.

If the current request is a transaction request and the USB bus time is enough for this transaction, USBHS will employ SIE to generate this transaction on USB bus.

When the required bus time by the current request is not enough in the current frame, if this is a periodic request, USBHS stops the processing of periodic queue and starts to process non-periodic request. If this is a non-periodic queue the USBHS will stop to process any queue and wait until the end of current frame.

**LPM**

This addendum for USB defines power management states (LPM states) and mechanisms to affect state changes that are used by hosts and hubs to efficiently manage bus and system power. LPM simply adds a new feature and bus state sleep state (L1) that co-exists with the USB2.0 defined suspend (L2)/resume.

L1 is similar to L2 but supports finer granularity in use. Entry to L1 is started by a request to a hub or host port to transition to L1. A LPM transaction is sent to the downstream device. The requested transition can only occur if the device response with an ACK handshake. Exit from L1 is via remote wake, resume signaling, reset signaling or disconnect. Either the host or device can initiate resume signaling when in L1. Although the signaling levels of resume are the same as L2, the duration of the signaling and transitional latencies associated with the L1 to L0 (active state) transition are much shorter.

### 49.5.3. USB device function

**USB Device Connection**

In device mode USBHS stays at power-off state after initialization. After connected to a USB host with 5V power supply present on VBUS pin or setting VDEN bit in USBHS_GCCFG register, USBHS enters into powered state. USBHS begins to switch on the pull-up resistor on DP line, host side will detect a connection event.

**Reset and Speed-Identification**

The USB host always starts a USB reset after it detects a device connection, USBHS in device mode will trigger a reset interrupt for software after it detects the reset event on USB bus.

If the maximum supported speed is configured to Full-Speed (DS[1:0] = 01 in USBHS_DCFG

register), USBHS will operate as a Full-Speed device. If the maximum supported speed is configured to High-Speed (DS[1:0] = 00 in USBHS_DCFG register), USBHS device tries to start a speed-identification (a chirp handshake described in USB 2.0 protocol) with host during reset sequence. If the chirp handshake with host successes, the device enters High-Speed mode, otherwise, remains at Full-Speed mode.

After reset sequence, speed-identification process completes, USBHS triggers an ENUMF interrupt in USBHS_GINTF register and reports current enumerated device speed in ES bits in USBHS_DSTAT register. If software want to implement a High-Speed device, it must wait ENUMF interrupt first, then read the ES[1:0] bits to get the speed-identification result.

As required by USB 2.0 protocol, USBHS doesn't support Low-Speed in device mode.

**Suspend and Wake-up**

A USB device will enter into suspend state after the USB bus stays at IDLE state and has no change on data lines for 3ms. When USB device is in suspend state, software can switch off most of its clock to save power. The USB host is able to wake up the suspended device by generating a resume signal on USB bus. USBHS is able to detect the resume signal and triggers the WKUPIF flag in USBHS_GINTF register and the USBHS wake up interrupt.

In suspend mode, USBHS is also able to remote wake-up the USB bus. Software may set RWKUP bit in USBHS_DCTL register to sends a remote-wake-up signal, and if remote-wake up is supported in USB host, the host will begin to send resume signal on USB bus.

**Soft Disconnection**

USBHS supports soft disconnection. After the device is power on, USBHS will switch on the pull-up resistor on DP line and this will cause the host to detect the connection. Then, software is able to force a disconnection by setting the SD bit in USBHS_DCTL register. After the SD bit is set, if the current device speed is High-Speed, USBHS will first return back to Full-Speed device and then switch off the pull-up resistor on DP line, and if current speed is Full-Speed, USBHS will directly switch off the pull-up resistor. This will cause USB host to detect a disconnection on USB bus.

**SOF tracking**

When USBHS receives a SOF packet from USB bus, it triggers a SOF interrupt and begins to count the bus time by using local USB clock. The frame number of the current frame is reported in FNRSOF[13:0] in USBHS_DSTAT register. When the USB bus time reaches EOF1 or EOF2 point (End of Frame, described in USB 2.0 protocol), USBHS will trigger a interrupt EOPFIF in USBHS_GINTF register. Software is able to use these flags and registers to get current bus time and position information.

**BCD**

Charging port detection (BCD) described in Battery Charging Specification Revision 1.2 is supported. In order for PD (portable device) to determine how much current it is allowed to draw from an upstream USB port, there need to be mechanisms that allow the PD to distinguish between a Standard Downstream Port and a Charging Port.

In BCD mechanisms, USB VBUS detection (VD), data contact detection (DCD), primary detection (PD) and secondary detection (SD) are included. The control and configuration bits about BCD is reported in USBHS_GCCFG register.

### 49.5.4. OTG function overview

USBHS supports OTG function described in OTG protocol 1.3/2.0. OTG function includes SRP and HNP protocols.

**A-Device and B-Device**

A-Device is an OTG capable USB device with a Standard-A or Micro-A plug inserted into its receptacle. The A-Device supplies power for VBUS and it is host at the start of a session. B-Device is an OTG capable USB device with a Standard-B, Micro-B or Mini-B plug inserted into its receptacle, or a captive cable ending in a Standard-A plug. The B-Device is a peripheral at the start of a session. USBHS uses the voltage level of ID pin to judge A-Device or B-Device. The ID status is reported in IDPS bit in USBHS_GOTGCS register. For the details of states transfer between A-Device and B-Device, please refer to OTG 1.3/2.0 protocol.

**HNP**

The Host Negotiation Protocol (HNP) allows the host function to be transferred between two directly connected On-The-Go devices and eliminates the need for a user to switch the cable connections in order to allow a change in control of communications between the devices. HNP will typically be initiated by the user or an application on the On-The-Go B-device. HNP may only be implemented through the Micro-AB receptacle on a device.

Since On-The-Go devices have a Micro-AB receptacle, an On-The-Go device can default to being either Host or Peripheral, depending upon which type of plug (Micro-A plug for Host, Micro-B plug for Peripheral) is inserted. By utilizing the Host Negotiation Protocol (HNP), an On-The-Go B-Device, which is the default Peripheral, may make a request to be Host. The process for this exchange of the role of Host is described in this section. This protocol eliminates the need for the user to swap the cable connection in order to change the roles of the connected devices.

When USBHS is in OTG A-Device host mode and it wants to give up its host role, it may first set PSP bit in USBHS_HPCS register to make the USB bus enter suspend status. Then, the B-device will enter suspend state after 3ms. If the B-Device wants to changes to host, software needs to set HNPREQ bit in USBHS_GOTGCS register and the USBHS will begin to perform HNP protocol on bus, and at last, the result of HNP is reported in HNPS bit in USBHS_GOTGCS register. Besides, software is always able to get the current role (host or peripheral) from COPM bit in USBHS_GINTF register.

**SRP**

The Session Request Protocol (SRP) allows a B-Device to request the A-Device to turn on VBUS and start a session. This protocol allows the A-Device, which may be battery powered,

to conserve power by turning VBUS off when there is no bus activity while still providing a means for the B-Device to initiate bus activity. As described in OTG protocol, an OTG device must compare VBUS voltage with several threshold values and the compare result is reported in ASV and BSV bits in USBHS_GOTGCS register.

Software may set SRPREQ bit in USBHS_GOTGCS register to start a SRP request when USBHS is in B-Device OTG mode and USBHS will generate a success flag SRPS in USBHS_GOTGCS register if the SRP request successes.

When USBHS is in OTG A-Device mode and it detects an SRP request from a B-Device, it sets a SESIF flag in USBHS_GINTF register. The software should prepare to switch on the 5V power supply for VBUS pin after it gets this flag.

**ADP**

Attach Detection Protocol (ADP) is a protocol that allows a local device to detect when a remote device has been attached or detached. The remote device can be any USB device. ADP operates by detecting the change in VBUS capacitance that occurs when two devices are attached or detached. The capacitance is detected by first discharging the VBUS line, and then measuring the time it takes VBUS to charge to a known voltage with a known current source. A change in capacitance is detected by looking for a change in the charge time.

Software may set bit ADPMEN, ADPEN and ENAPRB to perform ADP probe, and should perform at least one ADP probe cycle in order to obtain an initial value for TADP_RISE when an ADP-capable A-device or B-device is first powered up. For B-device, ADP sense can be performed by setting bit ENASNS. If RITM in USBHS_ADPCTL register changes, it shows that a remote device has been attached or detached.

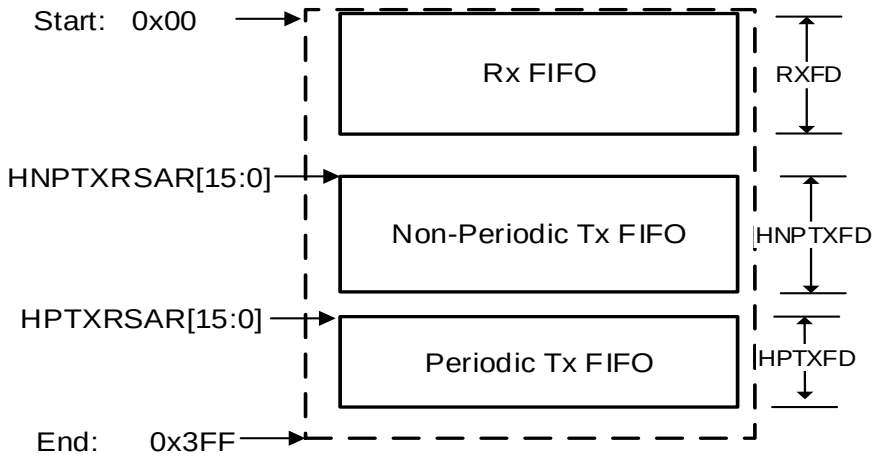
### 49.5.5. Data FIFO

The USBHS include a 4K bytes data FIFO to store packet data. The data FIFO is implemented by using an internal SRAM.

**Host Mode**

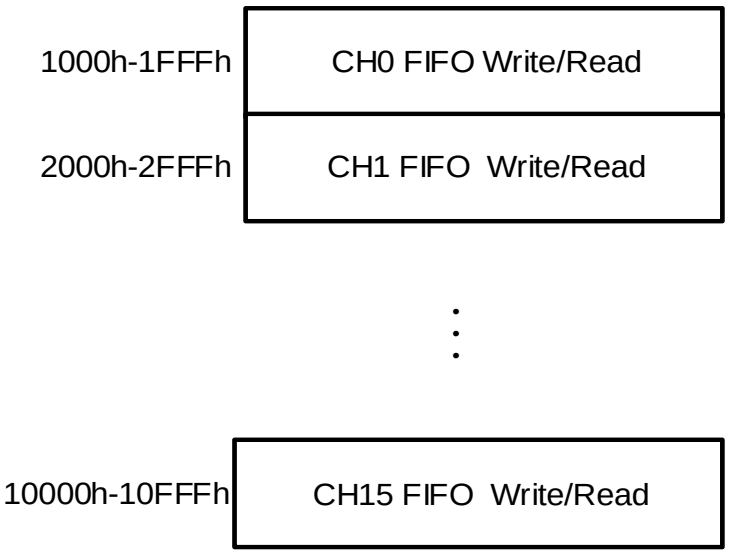
In host mode the data FIFO space is divided into 3 parts: Rx FIFO for received packet, Non-Periodic Tx FIFO for non-period transmission packet and Periodic Tx FIFO for periodic transmission packet. All IN channels shares the Rx FIFO for receiving packets. All the periodic OUT channels share the periodic Tx FIFO to transmit packets. All the non-periodic OUT channels share the non-Periodic FIFO for transmit packets. Software should configure the size and start offset of these data FIFOs by use these registers: USBHS_GRFLEN, USBHS_HNPTFLEN and USBHS_HPTFLEN. ***Figure 49-6. Host mode FIFO space in SRAM*** describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

**Figure 49-6. Host mode FIFO space in SRAM**




In DMA mode, DMA engine is responsible for packet data copy between system memory and the internal data FIFOs. In non-DMA mode the application needs to manually write packet data into or read packet from the data FIFOs. USBHS provides a special register area for software to write and read the internal data FIFO. ***Figure 49-7. Host mode FIFO access register map*** describes the register memory area for data FIFO access. The addresses in the figure are in term of byte. Each channel has its own FIFO access register space, although all Non-periodic channels share the same FIFO and all the Periodic channels share the same FIFO. This is important for USBHS to know the current pushed packet belongs to which channel. Rx FIFO is also able to be accessed by using USBHS_GRSTATR/USBHS_GRSTATP register.

**Figure 49-7. Host mode FIFO access register map**

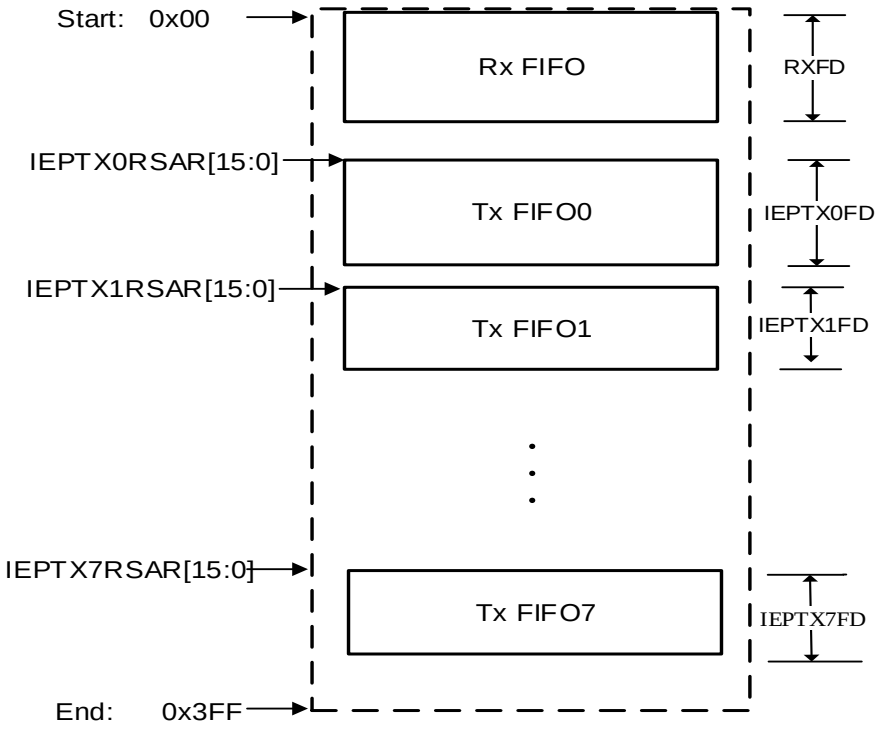



**Device mode**

In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 8 Tx FIFOs (one for each IN endpoint). All the OUT endpoints share the Rx FIFO for receiving packets.

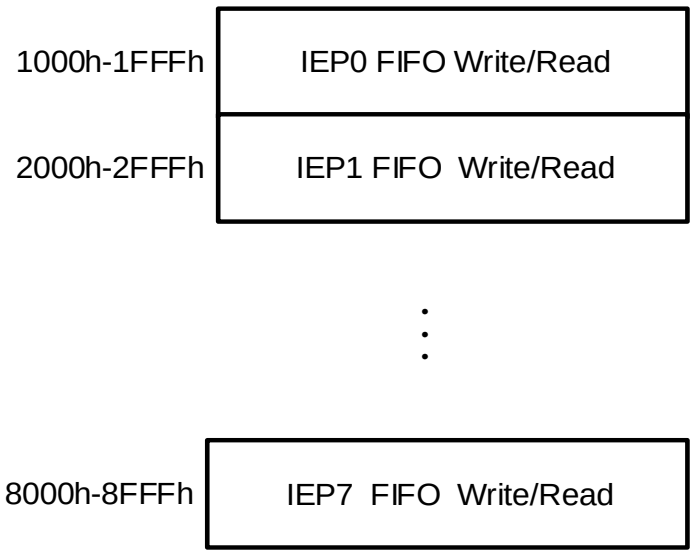
Software should configure the size and start offset of these data FIFOs by using USBHS_GRFLEN and USBHS_DIEPxTFLEN (x=0...7) registers. ***Figure 49-8. Device mode FIFO space in SRAM*** describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

**Figure 49-8. Device mode FIFO space in SRAM**




In DMA mode, DMA engine is responsible for packet data copy between system memory and the internal data FIFOs. In non-DMA mode the application needs to manually write packet data into or read packet from the data FIFOs. USBHS provides a special register area for software to write and read the internal data FIFO. ***Figure 49-9. Device mode FIFO access register map*** describes the register memory area for data FIFO access. The addresses in the figure are in term of byte. Each endpoint has its own FIFO access register space. Rx FIFO is also able to be accessed by using USBHS_GRSTATR/USBHS_GRSTATP register.

**Figure 49-9. Device mode FIFO access register map**




### 49.5.6. DMA function

This section describes the DMA scheduler and DMA engine in USBHS.

**DMA Requests and Scheduler**

DMA function is enabled by setting DMAEN bit in USBHS_GAHBCS register. When an IN/OUT channel or IN endpoint is properly configured and enabled, or the Rx FIFO is not empty, USBHS will generate DMA request. There is a DMA scheduler in USBHS responsible for responding to theses DMA requests.

There may be several requests simultaneously and the DMA scheduler arbitrates among these requests. These requests are sorted into 3 kinds: Rx FIFO DMA request, periodic transfer DMA requests and non-periodic transfer DMA requests. Rx FIFO DMA request takes the highest priority, and periodic transfer DMA requests take the medium priority, and non-periodic transfer DMA requests take the lowest priority when arbitration. DMA scheduler performs round-robin arbitration method within the periodic or non-periodic transfer DMA requests.

As is described above, DMA will automatically handle the Rx FIFO not empty event, so software should ignore the RXFNEIF flag in USBHS_GINTF register in DMA mode.

**DMA Engine**

Receive:

In host or device mode, once Rx FIFO DMA request gets arbitration, DMA engine begins to read a packet or a status entry from Rx FIFO. For data packet, DMA write the data into the specified system address configured in the HCHxDMAADDR register or DIEPxDMAADDR/DOEPxDMAADDR register. For status entry, DMA will generate the specified flags or interrupts on related channels or endpoints.

Host Transfer:

When a periodic or non-periodic IN channel DMA request gets arbitration, DMA writes IN request entries into the periodic or non-periodic request queue. After the desired IN transfers completes, or an AHB/USB bus error occurs, DMA halts the specified channel and generate TF and CH flags in USBHS_HCHxINTF register. The received packet during IN transfers copied into system memory after the Rx FIFO DMA request is generated, as described above.

When an OUT periodic or non-periodic channel DMA request gets arbitration, DMA reads packet data from system memory and writes to internal Tx FIFO. DMA always writes an OUT request entry into the request queue when it finishes a packet data copying. After the desired OUT transfers completes, or an AHB/USB bus error occurs, DMA halt the specified channel and generate TF and CH flags in USBHS_HCHxINTF register.

Device Transfer:

In device mode, when an IN endpoint DMA request gets arbitration, DMA reads packet data from system memory and writes to the endpoint's Tx FIFO. When USBHS gets an IN token on an IN endpoint, it transmits the packet copied by DMA engine.

### 49.5.7. Operation guide

This section describes the advised operation guide for USBHS.

#### Host mode

##### Global register initialization sequence

1. Program USBHS_GAHBCS register according to application's demand, such as: whether to enable DMA, burst type of DMA transfer and the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBHS_GUSBCS register according to application's demand, such as: the operation mode (host, device or OTG) and some parameters of OTG, ULPI and USB protocols.
3. Program USBHS_GCCFG register according to application's demand.
4. Program USBHS_GRFLEN, USBHS_HNPTFLEN_DIEP0TFLEN and USBHS_HPTFLEN registers to configure the data FIFOs according to application's demand.
5. Program USBHS_GINTEN register to enable Mode Fault and Host Port interrupt and set GINTEN bit in USBHS_GAHBCS register to enable global interrupt.
6. Program SPDFSLS bit in USBHS_HCTL register to select whether to limit the device speed to Full-Speed.
7. Program USBHS_HPCS register and set PP bit.
8. Wait for a device's connection, and once a device is connected, the connection interrupt

PCD in USBHS_HPCS register will be triggered. Then set PRST bit to perform a port reset. Wait for at least 10ms and then clear PRST bit.

9. Wait PEDC interrupt in USBHS_HPCS register and then read PE bit to ensure that the port is successfully enabled. Read PS[1:0] bits to get the connected device's speed and then program USBHS_HFT register if software want to change the SOF interval.

**Channel initialization and enable sequence**

1. Program USBHS_HCHxCTL register with desired transfer type, direction, packet size, etc. Ensure that CEN and CDIS bits keep cleared during configuration.
2. Program USBHS_HCHxINTEN register. Set the desired interrupt enable bits.
3. If DMA is enabled, program USBHS_HCHxDMAADDR register.
4. Program USBHS_HCHxLEN register. PCNT is the number of packets in a transfer and TLEN is the total bytes number of all the transmitted or received packets in a transfer.

   For OUT channel: If PCNT=1, the single packet's size is equal to TLEN. If PCNT>1, the former PCNT-1 packets are considered as max-packet-length packets whose size are defined by MPL field in USBHS_HCHxCTL register, and the last packet's size is calculated based on PCNT, TLEN and MPL. If software wants to send out a zero-length packet, it should program TLEN=0, PCNT=1.

   For IN channel: Because the application doesn't know the actual received data size before the IN transaction finishes, software may program TLEN as a maximum possible value supported by Rx FIFO.
5. Set CEN bit in USBHS_HCHxCTL register to enable the channel.

**Channel disable sequence**

Software can disable the channel by setting both CEN and CDIS bits at the same time. USBHS will generate a channel disable request entry in request queue after the register setting operation. When the request entry reaches to the top of request queue, it is processed by USBHS immediately:

For OUT channel, the specified channel will be disabled immediately. Then, a CH flag will be generated and the CEN and CDIS bits will be cleared by USBHS.

For IN channels, USBHS pushes a channel disable status entry into Rx FIFO. Software should then handle the Rx FIFO not empty event: read and pop this status entry, then, a CH flag will be generated and the CEN and CDIS bits will be cleared.

**IN transfers operation sequence with DMA disabled**

1. Initialize USBHS global registers.
2. Initialize the channel.
3. Enable the channel.

4. After the IN channel is enabled by software, USBHS generates a Rx request entry in the corresponding request queue.
5. When the Rx request entry reaches to the top of the request queue, USBHS begins to process this request entry. If bus time for the IN transaction indicated by the request entry is enough, USBHS starts the IN transaction on USB bus.
6. If the IN transaction finishes successfully (ACK handshake received), USBHS pushes the received data packet into the Rx FIFO and triggers ACK flag. Otherwise, the status flag (NAK) report the transaction result.
7. If the IN transaction described in step 5 is successful and PCNT is larger than 1 in step2, software should return to step 3 and continues to receive the remaining packets. If the IN transaction described in step 5 is not successful, software should return to step 3 to re-receive the packet again.
8. After all the transactions in a transfer are successful received on USB bus, USBHS push a TF status entry into the Rx FIFO on top of the last packet data. After software reads and pops all the received data packet, and at last, the TF status entry, USBHS generates TF flag to indicate that the transfer successfully finishes.
9. Disable the channel. Now the channel is in IDLE state and is ready for other transfers.

**IN transfers operation sequence with DMA enabled**

1. Initialize USBHS global registers.
2. Initialize and enable the channel.
3. After the IN channel is enabled by software, USBHS begins to generate Rx request entry in the corresponding request queue.
4. USBHS processes the request entries in request queue one by one and perform the indicated IN transactions on USB bus.
5. When a IN transaction gets a NAK handshake, the DMA is able to re-send IN tokens automatically until that USBHS get the desired number of packets.
6. After USBHS gets the desired number of packets specified by PCNT in USBHS_HCHxLEN register, USBHS generates TF and CH flags to indicate that the transfer successfully finishes and the channel is disabled. If USB bus error or DMA write error occurs during these transactions, DMA will trigger related error flags, stops the processing for this channel, disable this channel and at last, trigger the CH flag.

**Note:** In DMA mode, software should not enable or process the RXFNEIF interrupt because the DMA will automatically process the Rx FIFO.

**OUT transfers operation sequence with DMA disabled**

1. Initialize USBHS global registers.
2. Initialize and enable the channel.

3. Write a packet into the channel's Tx FIFO (Periodic Tx FIFO or non-periodic Tx FIFO). After the whole packet data is written into the FIFO, USBHS generates a Tx request entry in the corresponding request queue and decrease the TLEN field in USBHS_HCHxLEN register with the written packet's size.

4. When the request entry reaches to the top of the request queue, USBHS begins to process this request entry. If bus time for the transaction indicated by the request entry is enough, USBHS starts the OUT transaction on USB bus.

5. When the OUT transaction indicated by the request entry finishes on USB bus, PCNT in USBHS_HCHxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flag (NAK) report the transaction result.

6. If the OUT transaction described in step 5 is successful and PCNT is larger than 1 in step2, software should return to step 3 and continues to send the remaining packets. If the OUT transaction described in step 5 is not successful, software should return to step 3 to resend the packet again.

7. After all the transactions in a transfer are successful sent on USB bus, USBHS generates TF flag to indicate that the transfer successfully finishes.

8. Disable the channel. Now the channel is in IDLE state and is ready for other transfers.

**OUT transfers operation sequence with DMA enabled**

1. Initialize USBHS global registers.

2. Initialize and enable the channel.

3. DMA in USBHS begins to fetch packets from the address specified by DMAADDR in USBHS_HCHxDMAADDR register and write them into the channel's Tx FIFO (Periodic Tx FIFO or non-periodic Tx FIFO). Each time a whole packet data is written into the FIFO, USBHS generates a Tx request entry in the corresponding request queue and decrease the TLEN field in USBHS_HCHxLEN register with the written packet's size.

4. USBHS processes the request entries in request queue one by one and sends out the indicated transactions on USB bus.

5. When a transaction gets a NAK or NYET handshake, the DMA is able to re-fetch and re-send the packet as well as perform PING protocol automatically.

6. If all the transactions are successful sent on USB bus, USBHS generates TF and CH flags to indicate that the transfer successfully finishes and the channel is disabled. If USB bus error or DMA fetch error occurs during these transactions, DMA will trigger related error flags, stops the processing for this channel, disable this channel and at last, trigger the CH flag.

**Note:** In DMA mode, software should not enable or process the RXFNEIF interrupt because the DMA will automatically process the Rx FIFO.

### Device mode

#### Global register initialization sequence

1. Program USBHS_GAHBCS register according to application's demand, such as: whether to enable DMA, burst type of DMA transfer and the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBHS_GUSBCS register according to application's demand, such as: the operation mode (host, device or OTG) and some parameters of OTG, ULPI and USB protocols.
3. Program USBHS_GCCFG register according to application's demand.
4. Program USBHS_GRFLEN, USBHS_HNPTFLEN_DIEP0TFLEN and USBHS_DIEPxTFLEN registers to configure the data FIFOs according to application's demand.
5. Program USBHS_GINTEN register to enable Mode Fault, Suspend, SOF, Enumeration Done and USB Reset interrupt and then, set GINTEN bit in USBHS_GAHBCS register to enable global interrupt.
6. Program USBHS_DCFG register according to application's demand, such as the device speed and device address, etc.
7. After the device is connected to a host, the host will perform port reset on USB bus and this will trigger the RST interrupt in USBHS_GINTF register.
8. Wait for ENUMF interrupt in USBHS_GINTF register and then read ES[1:0] bits in USBHS_DSTAT register to get the current enumerated device speed.

#### Endpoint initialization and enable sequence

1. Program USBHS_DIEPxCTL or USBHS_DOEPxCTL register with desired transfer type, packet size, etc.
2. Program USBHS_DIEPINTEN or USBHS_DOEPINTEN register. Set the desired interrupt enable bits.
3. If DMA is enabled, program USBHS_DIEPxDMAADDR or USBHS_DOEPxDMAADDR register.
4. Program USBHS_DIEPxLEN or USBHS_DOEPxLEN register. PCNT is the number of packets in a transfer and TLEN is the total bytes number of all the transmitted or received packets in a transfer.

   For IN endpoint：If PCNT=1, the single packet's size is equal to TLEN. If PCNT>1, the former PCNT-1 packets are considered as max-packet-length packets whose size are defined by MPL field in USBHS_DIEPxCTL register, and the last packet's size is calculated based on PCNT, TLEN and MPL. If software wants to send out a zero-length packet, it should program TLEN=0, PCNT=1.

For OUT endpoint：Because the application doesn't know the actual received data size before the OUT transaction finishes, software may program TLEN as a maximum possible value supported by Rx FIFO.

5. Set EPEN bit in USBHS_DIEPxCTL or USBHS_DOEPxCTL register to enable the endpoint.

**Endpoint disable sequence**

Software can disable the endpoint anytime when clearing the EPEN bit in USBHS_DIEPxCTL or USBHS_DOEPxCTL register.

**IN transfers operation sequence with DMA disabled**

1. Initialize USBHS global registers.
2. Initialize and enable the IN endpoint.
3. Write packets into the endpoint's Tx FIFO. Each time a data packet is written into the FIFO, USBHS decreases the TLEN field in USBHS_DIEPxLEN register with the written packet's size.
4. When an IN token is received, USBHS transmit the data packet, and after the transaction finishes on USB bus, PCNT in USBHS_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags report the transaction result.
5. After all the data packets in a transfer are successful sent on USB bus, USBHS generates TF flag to indicate that the transfer successfully finishes and disable the IN endpoint.

**IN transfers operation sequence with DMA enabled**

1. Initialize USBHS global registers.
2. Initialize and enable the IN endpoint.
3. DMA in USBHS begins to fetch packets from the address specified by DMAADDR in USBHS_DIEPxDMAADDR register and write them into the IN endpoint's Tx FIFO. Each time a whole packet data is written into the FIFO, USBHS decreases the TLEN field in USBHS_DIEPxLEN register with the written packet's size.
4. When an IN token is received, USBHS transmit the data packet, and after the transaction finishes on USB bus, PCNT in USBHS_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags report the transaction result.
5. If all the transactions are successful sent on USB bus, USBHS generates TF and EPDIS flags to indicate that the transfer successfully finishes and the endpoint is disabled. If USB bus error or DMA fetch error occurs during these transactions, DMA will trigger related error flags.

**Note:** In DMA mode, software should not enable or process the RXFNEIF interrupt because

the DMA will automatically process the Rx FIFO.

**OUT transfers operation sequence with DMA disabled**

1. Initialize USBHS global registers.
2. Initialize the endpoint and enable the endpoint.
3. When an OUT token is received, USBHS receive the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBHS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBHS_DOEPxLEN register is decreased by 1 and the ACK flag is triggered, otherwise, the status flags report the transaction result.
4. After all the data packets in a transfer are successful received on USB bus, USBHS push a TF status entry into the Rx FIFO on top of the last packet data. After software reads and pops all the received data packet, and at last, the TF status entry, USBHS generates TF flag to indicate that the transfer successfully finishes and disable the OUT endpoint.

**OUT transfers operation sequence with DMA enabled**

1. Initialize USBHS global registers.
2. Initialize and enable the OUT endpoint.
3. When an OUT token received, USBHS receive the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBHS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBHS_DOEPxLEN register is decreased by 1 and the ACK flag is triggered, otherwise, the status flags report the transaction result.
4. If all the transactions are successful received on USB bus, USBHS generates TF and EPDIS flags to indicate that the transfer successfully finishes and the endpoint is disabled. If USB bus error or DMA write error occurs during these transactions, DMA will trigger related error flags.

**Note:** In DMA mode, software should not enable or process the RXFNEIF interrupt because the DMA will automatically process the Rx FIFO.

## 49.6. Interrupts

USBHS has four interrupts: global interrupt, wake-up interrupt, endpoint1 IN interrupt and endpoint1 OUT interrupt.

Global interrupt is the main interrupt software should process, the source flags of the global interrupt are readable in USBHS_GINTF register and listed in the following ***Table 49-3. USBHS global interrupt***.

**Table 49-3. USBHS global interrupt**

| Interrupt Flag | Description | Operation Mode |
|---|---|---|
| SEIF | Session interrupt | Host or device mode |
| DISCIF | Disconnect interrupt flag | Host Mode |
| IDPSC | ID pin status change | Host or device mode |
| LPMIF | LPM interrupt flag | Host or device mode |
| PTXFEIF | Periodic Tx FIFO empty interrupt flag | Host Mode |
| HCIF | Host channels interrupt flag | Host Mode |
| HPIF | Host port interrupt flag | Host Mode |
| ISOONCIF/PXNCIF | Periodic transfer Not Complete Interrupt flag / Isochronous OUT transfer Not Complete Interrupt Flag | Host or device mode |
| ISOINCIF | Isochronous IN transfer Not Complete Interrupt Flag | Device mode |
| OEPIF | OUT endpoint interrupt flag | Device mode |
| IEPIF | IN endpoint interrupt flag | Device mode |
| EOPFIF | End of periodic frame interrupt flag | Device mode |
| ISOOPDIF | Isochronous OUT packet dropped interrupt flag | Device mode |
| ENUMF | Enumeration finished | Device mode |
| RST | USB reset | Device mode |
| SP | USB suspend | Device mode |
| ESP | Early suspend | Device mode |
| GONAK | Global OUT NAK effective | Device mode |
| GNPINAK | Global IN Non-Periodic NAK effective | Device mode |
| NPTXFEIF | Non-Periodic Tx FIFO empty interrupt flag | Host Mode |
| RXFNEIF | Rx FIFO non-empty interrupt flag | Host or device mode |
| SOF | Start of frame | Host or device mode |
| OTGIF | OTG interrupt flag | Host or device mode |
| MFIF | Mode fault interrupt flag | Host or device mode |

Wake up interrupt is able to be triggered when USBHS is in suspend state, even when the USBHS's clocks are stopped. The source of the wake up interrupt is WKUPIF bit in USBHS_GINTF register.

Endpoint 1 IN/OUT interrupts are two special interrupts for endpoint 1. Application can use these two interrupts to make a quick response to the events on endpoint 1. The two interrupts are individually enabled by USBHS_DEP1INT register. And the source of these two interrupts also come from USBHS_DIEP1INTF and USBHS_DOEP1INTF registers, but the enable bits for these flags to generate Endpoint 1 IN/OUT interrupts are in USBHS_DIEP1INTEN and USBHS_DOEP1INTEN registers.

# 49.7. Register definition

USBHS0 base address: 0x4004 0000

USBHS1 base address: 0x4008 0000

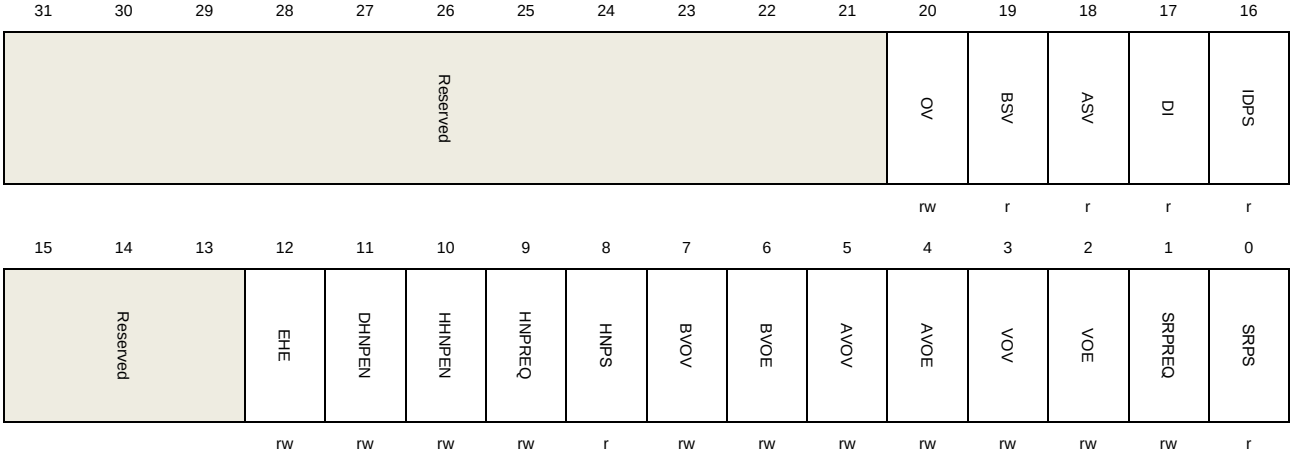
## 49.7.1. USBHS global registers

### Global OTG control and status register (USBHS_GOTGCS)

Address offset: 0x0000

Reset value: 0x0000 0800

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | OV | Select OTG version<br>0: Version 1.3 is selected. Data line pulsing and VBUS pulsing are supported in SRP<br>1: Version 2.0 is selected. Only data line pulsing is supported in SRP |
| 19 | BSV | B-Session Valid (described in OTG protocol).<br>0: Vbus voltage level of a OTG B-device is below VBSESSVLD<br>1: Vbus voltage level of a OTG B-Device is not below VBSESSVLD<br>**Note:** Only accessible in OTG B-Device mode. |
| 18 | ASV | A-session valid<br>A-host mode transceiver status.<br>0: Vbus voltage level of a OTG A-device is below VASESSVLD<br>1: Vbus voltage level of a OTG A-device is not below VASESSVLD<br>The A-device is as host default at the start of a session.<br>**Note:** Only accessible in OTG A-Device mode. |

| | | |
|---|---|---|
| 17 | DI | Debounce interval<br>Debounce interval of a detected connection.<br>0: Indicates the long debounce interval, when a plug-on and connection occur on USB bus<br>1: Indicates the short debounce interval, when a soft connection is used in HNP protocol.<br>**Note:** Only accessible in host mode. |
| 16 | IDPS | ID pin status<br>Voltage level of connector ID pin<br>0: USBHS is in A-device mode<br>1: USBHS is in B-device mode<br>**Note:** Accessible in both device and host modes. |
| 15:13 | Reserved | Must be kept at reset value |
| 12 | EHE | Embedded host enable<br>0: OTG A-device state is selected<br>1: Embedded host state is selected |
| 11 | DHNPEN | Device HNP enable<br>Enable the HNP function of a B-device. If this bit is cleared, USBHS doesn't start HNP protocol when application set HNPREQ bit in USBHS_GOTGCS register.<br>0: HNP function is not enabled.<br>1: HNP function is enabled<br>**Note:** Only accessible in device mode. |
| 10 | HHNPEN | Host HNP enable<br>Enable the HNP function of an A-device. If this bit is cleared, USBHS doesn't response to the HNP request from B-device.<br>0: HNP function is not enabled.<br>1: HNP function is enabled<br>**Note:** Only accessible in host mode. |
| 9 | HNPREQ | HNP request<br>This bit is set by software to start a HNP on the USB. Software can clear this bit when HNPEND bit in USBHS_GOTGINTF register is set, by writing zero to it, or clearing the HNPEND bit in USBHS_GOTGINTF register.<br>0: Don't send HNP request<br>1: Send HNP request<br>**Note:** Only accessible in device mode. |
| 8 | HNPS | HNP successes<br>This bit is set by the core when HNP successes and cleared when HNPREQ bit is set.<br>0: HNP fails<br>1: HNP successes |

| | | |
|---|---|---|
| | | **Note:** Only accessible in device mode. |
| 7 | BVOV | Override value of B-peripheral session valid<br>0: B-peripheral session valid value is 0 when BVOE = 1<br>1: B-peripheral session valid value is 1 when BVOE = 1<br>**Note:** Only accessible in device mode. |
| 6 | BVOE | Override enable of B-peripheral session valid<br>0: Override is disable. Internally B-peripheral session valid received from PHY is selected<br>1: Override is enable. Internally B-peripheral session valid received from PHY is overridden with BVOV<br>**Note:** Only accessible in device mode. |
| 5 | AVOV | Override value of A-peripheral session valid<br>0: A-peripheral session valid value is 0 when AVOE = 1<br>1: A-peripheral session valid value is 1 when AVOE = 1<br>**Note:** Only accessible in host mode. |
| 4 | AVOE | Override enable of A-peripheral session valid<br>0: Override is disable. Internally A-peripheral session valid received from PHY is selected<br>1: Override is enable. Internally A-peripheral session valid received from PHY is overridden with AVOV<br>**Note:** Only accessible in host mode. |
| 3 | VOV | Override value of VBUS valid<br>0: VBUS valid value is 0 when VOE = 1<br>1: VBUS valid value is 1 when VOE = 1<br>**Note:** Only accessible in host mode. |
| 2 | VOE | Override enable of VBUS valid<br>0: Override is disable. Internally VBUS valid received from PHY is selected<br>1: Override is enable. Internally VBUS valid received from PHY is overridden with VOV<br>**Note:** Only accessible in host mode. |
| 1 | SRPREQ | SRP request<br>This bit is set by software to start a SRP on the USB. Software can clear this bit when SRPEND bit in USBHS_GOTGINTF register is set, by writing zero to it, or clearing the SRPEND bit in USBHS_GOTGINTF register.<br>0: No session request<br>1: Session request<br>**Note:** Only accessible in device mode. |
| 0 | SRPS | SRP success<br>This bit is set by the core when SRP successes and cleared when SRPREQ bit is set. |

0: SRP fails

1: SRP successes

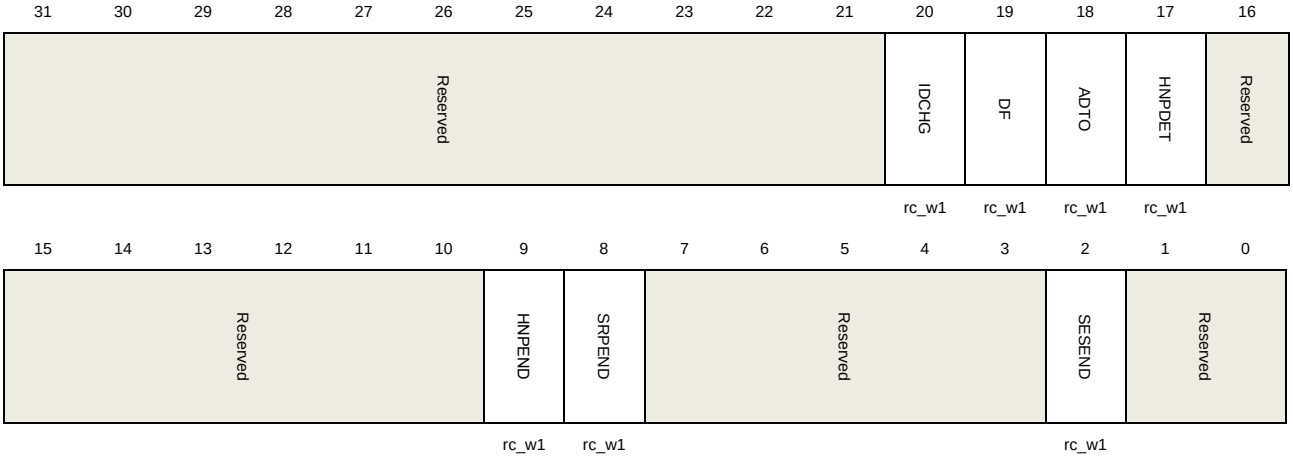
**Note:** Only accessible in device mode.

### Global OTG interrupt flag register (USBHS_GOTGINTF)

Address offset: 0x0004

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20 | IDCHG | There is a change in the value of ID input |
| 19 | DF | Debounce finish<br>Set by USBHS when the debounce during device connection is done.<br>**Note:** Only accessible in host mode. |
| 18 | ADTO | A-device timeout<br>Set by USBHS when the A-device's waiting for a B-device' connection has timed out.<br>**Note:** Accessible in both device and host modes. |
| 17 | HNPDET | Host negotiation request detected<br>Set by USBHS when A-device detects a HNP request.<br>**Note:** Accessible in both device and host modes. |
| 16:10 | Reserved | Must be kept at reset value. |
| 9 | HNPEND | HNP end<br>Set by the core when a HNP ends. Software should read the HNPS in USBHS_GOTGCS register to get the result of HNP.<br>**Note:** Accessible in both device and host modes. |
| 8 | SRPEND | SRPEND |

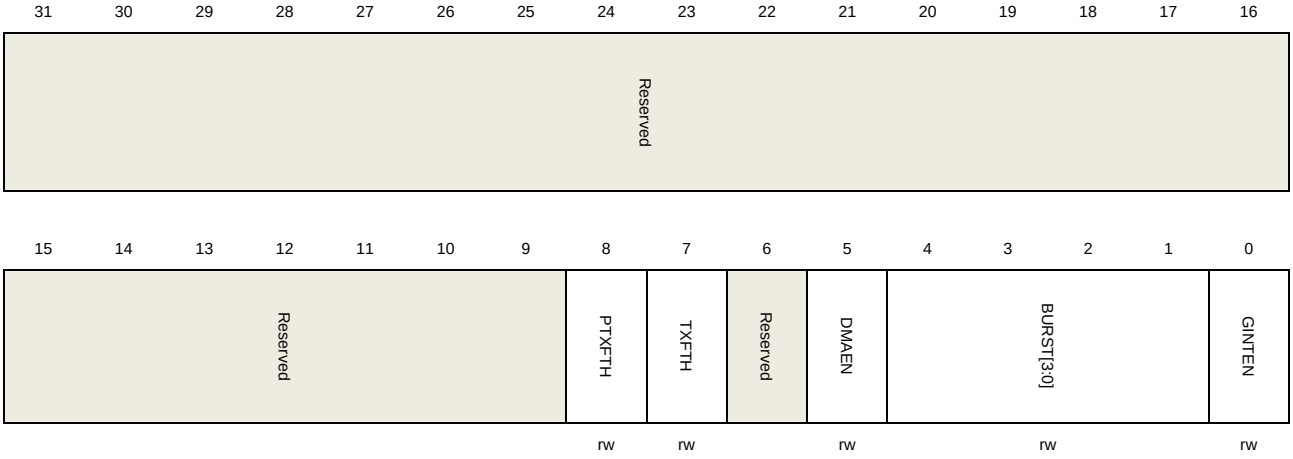
| | | |
|---|---|---|
| | | Set by the core when a SRP ends. Software should read the SRPS in USBHS_GOTGCS register to get the result of SRP.<br>**Note:** Accessible in both device and host modes. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | SESEND | Session end<br>Set by the core when VBUS voltage is below Vb_ses_vld. |
| 1:0 | Reserved | Must be kept at reset value. |

### Global AHB control and status register (USBHS_GAHBCS)

Address offset: 0x0008

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:9 | Reserved | Must be kept at reset value. |
| 8 | PTXFTH | Periodic Tx FIFO threshold<br>0: PTXFEIF will be triggered when the periodic transmit FIFO is half empty<br>1: PTXFEIF will be triggered when the periodic transmit FIFO is completely empty<br>**Note:** Only accessible in host mode. |
| 7 | TXFTH | Tx FIFO threshold<br>**Device mode:**<br>0: TXFEIF will be triggered when the IN endpoint transmit FIFO is half empty<br>1: TXFEIF will be triggered when the IN endpoint transmit FIFO is completely empty<br>**Host mode:**<br>0: NPTXFEIF will be triggered when the non-periodic transmit FIFO is half empty<br>1: NPTXFEIF will be triggered when the non-periodic transmit FIFO is completely empty |
| 6 | Reserved | Must be kept at reset value. |

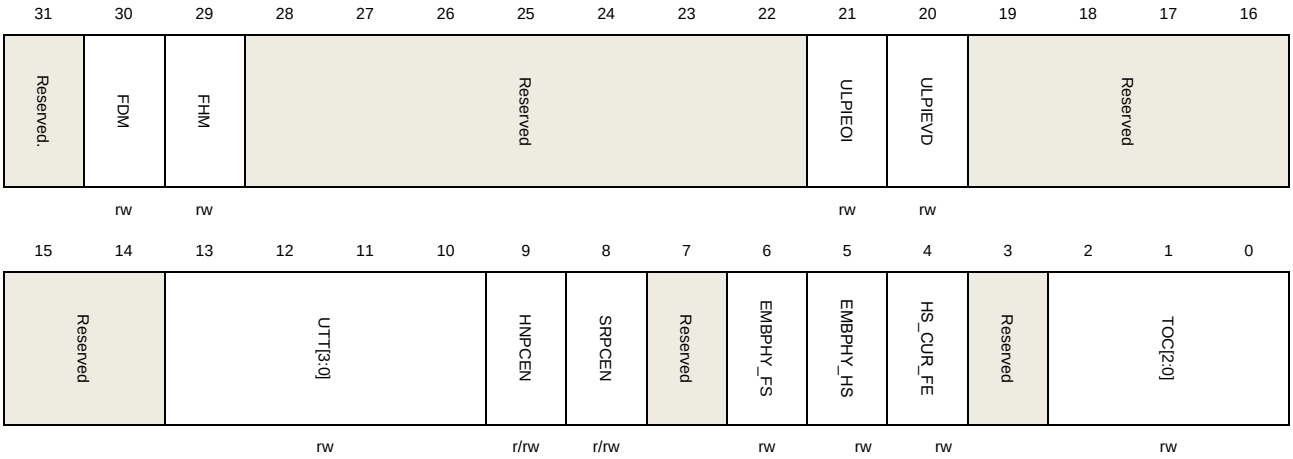
| | | |
|---|---|---|
| 5 | DMAEN | DMA function Enable<br>0: DMA function is disabled<br>1: DMA function is enabled |
| 4:1 | BURST[3:0] | The AHB burst type used by DMA<br>0000: Single<br>0001: INCR<br>0011: INCR4<br>0101: INCR8<br>0111: INCR16 |
| 0 | GINTEN | Global interrupt enable<br>0: Global interrupt is not enabled.<br>1: Global interrupt is enabled.<br>**Note:** Accessible in both device and host modes. |

### Global USB control and status register (USBHS_GUSBCS)

Address offset: 0x000C

Reset value: 0x0000 1400

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30 | FDM | Force device mode<br>Setting this bit will force the core to device mode irrespective of the USBHS ID input pin.<br>0: Normal mode<br>1: Device mode<br>The application must wait at least 25 ms for the change taking effect after setting the force bit.<br>**Note:** Accessible in both device and host modes. |

| | | |
|---|---|---|
| 29 | FHM | Force host mode<br>Setting this bit will force the core to host mode irrespective of the USBHS ID input pin.<br>0: Normal mode<br>1: Host mode<br>The application must wait at least 25 ms for the change taking effect after setting the force bit.<br>**Note:** Accessible in both device and host modes. |
| 28:22 | Reserved | Must be kept at reset value. |
| 21 | ULPIEOI | ULPI external over-current indicator<br>ULPI PHY uses this bit to decide whether to use internal or external over-current indicator. This bit only takes effect when external ULPI PHY is used (EMBPHY_FS and EMBPHY_HS bits in this register are both 0).<br>0: ULPI PHY uses internal over-current indicator<br>1: ULPI PHY uses external over-current indicator |
| 20 | ULPIEVD | ULPI external VBUS driver<br>ULPI PHY uses this bit to decide whether VBUS is driven by ULPI PHY or by external power supply. This bit only takes effect when external ULPI PHY is used (EMBPHY_FS and EMBPHY_HS bits in this register are both 0).<br>0: VBUS is driven by ULPI PHY<br>1: VBUS is driven by external power supply |
| 19:14 | Reserved | Must be kept at reset value. |
| 13:10 | UTT[3:0] | USB turnaround time<br>Turnaround time in PHY clocks.<br>**Note:** Only accessible in device mode. |
| 9 | HNPCEN | HNP capability enable<br>Controls whether the HNP capability is enabled<br>0: HNP capability is disabled<br>1: HNP capability is enabled<br>**Note:** Accessible in both device and host modes. |
| 8 | SRPCEN | SRP capability enable<br>Controls whether the SRP capability is enabled<br>0: SRP capability is disabled<br>1: SRP capability is enabled<br>**Note:** Accessible in both device and host modes. |
| 7 | Reserved | Must be kept at reset value. |
| 6 | EMBPHY_FS | Embedded FS PHY selected<br>0: Embedded FS PHY is disabled<br>1: Embedded FS PHY is enabled<br>**Note:** This bit can only be set when EMBPHY_HS is set to 0. Accessible in both |

| | | |
|---|---|---|
| | | device and host modes. |
| 5 | EMBPHY_HS | Embedded HS PHY selected<br>0: Embedded HS PHY is disabled<br>1: Embedded HS PHY is enabled<br>**Note:** This bit can only be set when EMBPHY_FS is set to 0. Accessible in both device and host modes. |
| 4 | HS_CUR_FE | HS current software enable<br>0: Release the HS mode TX current enable<br>1: Force HS mode TX current enable |
| 3 | Reserved | Must be kept at reset value. |
| 2:0 | TOC[2:0] | Timeout calibration<br>USBHS always uses time-out value required in USB 2.0 when waiting for a packet. Application may use TOC[2:0] to add the value is in terms of PHY clock. (The frequency of PHY clock is decided by which PHY is used: 48MHZ with internal embedded PHY and 60MHz with external ULPI PHY.) |

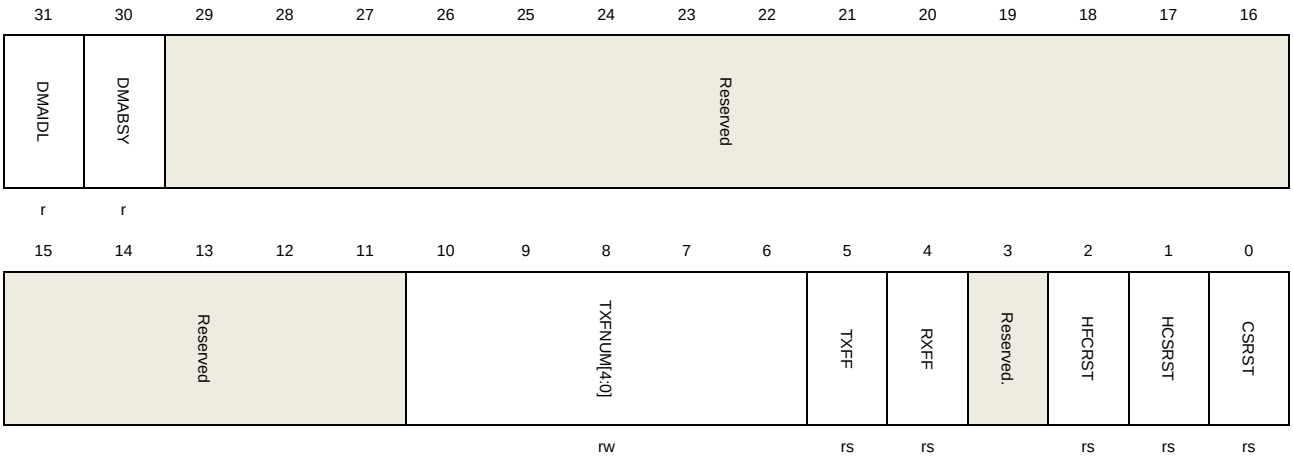
### Global reset control register (USBHS_GRSTCTL)

Address offset: 0x0010

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | DMAIDL | DMA Idle state<br>This bit reports that whether DMA is in IDLE state or not.<br>0: DMA is not IDLE<br>1: DMA is IDLE<br>**Note:** Accessible in both device and host modes. |

| | | |
|---|---|---|
| 30 | DMABSY | DMA Busy<br>This bit reports that whether DMA is busy.<br>0: DMA is not busy<br>1: DMA is busy<br>**Note:** Accessible in both device and host modes. |
| 29:11 | Reserved | Must be kept at reset value. |
| 10:6 | TXFNUM[4:0] | Tx FIFO number<br>Indicates which Tx FIFO will be flushed when TXFF bit in the same register is set.<br>Host Mode:<br>00000: Only non-periodic Tx FIFO is flushed<br>00001: Only periodic Tx FIFO is flushed<br>1XXXX: Both periodic and non-periodic Tx FIFOs are flushed<br>Other: Non data FIFO is flushed<br>Device Mode:<br>00000: Only Tx FIFO0 is flushed<br>00001: Only Tx FIFO1 is flushed<br>…<br>00111: Only Tx FIFO7 is flushed<br>1XXXX: All Tx FIFOs are flushed<br>Other: Non data FIFO is flushed |
| 5 | TXFF | Tx FIFO flush<br>Application sets this bit to flush data Tx FIFOs and TXFNUM[4:0] bits decide the FIFO number to be flushed. Hardware automatically clears this bit after the flush process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBHS.<br>**Note:** Accessible in both device and host modes. |
| 4 | RXFF | Rx FIFO flush<br>Application sets this bit to flush data Rx FIFO. Hardware automatically clears this bit after the flush process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBHS.<br>**Note:** Accessible in both device and host modes. |
| 3 | Reserved | Must be kept at reset value. |
| 2 | HFCRST | Host frame counter reset<br>Set by the application to reset the frame number counter in USBHS. After this bit is set, the frame number of the following SOF returns to 0. Hardware automatically clears this bit after the reset process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBHS.<br>**Note:** Only accessible in host mode. |
| 1 | HCSRST | HCLK soft reset<br>Set by the application to reset AHB clock domain circuit.<br>Hardware automatically clears this bit after the reset process completes. After |

setting this bit, application should wait until this bit is cleared before any other operation on USBHS.

**Note:** Accessible in both device and host modes.

0 CSRST Core soft reset

Resets the AHB and USB clock domains circuits, as well as most of the registers.

### Global interrupt flag register (USBHS_GINTF)

Address offset: 0x0014

Reset value: 0x0400 0021

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WKUPIF | SESIF | DISCIF | IDPSC | LPMIF | PTXFEIF | HCIF | HPIF | Reserved | | PXNCIF/ ISOONCIF | ISOINCIF | OEPIF | IEPIF | Reserved | |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | r | r | r | | | rc_w1 | rc_w1 | r | r | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EOPFIF | ISOOPDIF | ENUMF | RST | SP | ESP | Reserved | | GONAK | GNPINAK | NPTXFEIF | RXFNEIF | SOF | OTGIF | MFIF | COPM |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | | | r | r | r | r | rc_w1 | r | rc_w1 | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | WKUPIF | Wakeup interrupt flag<br>This interrupt is triggered when a resume signal (in device mode) or a remote wakeup signal (in host mode) is detected on the USB.<br>**Note:** Accessible in both device and host modes. |
| 30 | SESIF | Session interrupt flag<br>This interrupt is triggered when a SRP is detected (in A-Device mode) or $V_{BUS}$ becomes valid for a B- device (in B-Device mode).<br>**Note:** Accessible in both device and host modes. |
| 29 | DISCIF | Disconnect interrupt flag<br>This interrupt is triggered after a device disconnection.<br>**Note:** Only accessible in host mode. |
| 28 | IDPSC | ID pin status change<br>Set by the core when ID status changes.<br>**Note:** Accessible in both device and host modes. |
| 27 | LPMIF | LPM interrupt flag<br>In host mode, when device responds to LPM transaction with ACK, NYET or STALL, or when host has sent RECNT (USBHS_LPMCFG register) times LPM transaction, |

| | | |
|---|---|---|
| | | the interrupt is triggered.<br>In device mode, when device has received LPM transaction and responds with ACK, NYET or STALL, the interrupt is triggered. |
| 26 | PTXFEIF | Periodic Tx FIFO empty interrupt flag<br>This interrupt is triggered when the periodic transmit FIFO is either half or completely empty. The threshold is determined by the periodic Tx FIFO empty level bit (PTXFTH) in the USBHS_GAHBCS register.<br>**Note:** Only accessible in host mode. |
| 25 | HCIF | Host channels interrupt flag<br>Set by USBHS when one of the channels in host mode has raised an interrupt. Software should first read USBHS_HACHINT register to get the channel number, and then read the corresponding USBHS_HCHxINTF register to get the flags of the channel that cause the interrupt. This bit will be automatically cleared after the respective channel's flags which cause channel interrupt are cleared.<br>**Note:** Only accessible in host mode. |
| 24 | HPIF | Host port interrupt flag<br>Set by the core when USBHS detects that port status changes in host mode. Software should read USBHS_HPCS register to get the source of this interrupt. This bit will be automatically cleared after the flags that causing a port interrupt are cleared.<br>**Note:** Only accessible in host mode. |
| 23:22 | Reserved | Must be kept at reset value. |
| 21 | PXNCIF | Periodic transfer Not Complete Interrupt flag<br>USBHS sets this bit when there are periodic transactions for current frame not completed at the end of frame. (Host mode) |
| | ISOONCIF | Isochronous OUT transfer Not Complete Interrupt Flag<br>At the end of a periodic frame (defined by EOPFT bit in USBHS_DCFG), USBHS will set this bit if there are still isochronous OUT endpoints that not completed transactions. (Device Mode) |
| 20 | ISOINCIF | Isochronous IN transfer Not Complete Interrupt Flag<br>At the end of a periodic frame (defined by EOPFT[1:0] bits in USBHS_DCFG), USBHS will set this bit if there are still isochronous IN endpoints that not completed transactions. (Device Mode)<br>**Note:** Only accessible in device mode. |
| 19 | OEPIF | OUT endpoint interrupt flag<br>Set by USBHS when one of the OUT endpoints in device mode has raised an interrupt. Software should first read USBHS_DAEPINT register to get the device number, and then read the corresponding USBHS_DOEPxINTF register to get the flags of the endpoint that cause the interrupt. This bit will be automatically cleared after the respective endpoint's flags which cause this interrupt are cleared. |

| | | |
|---|---|---|
| | | **Note:** Only accessible in device mode. |
| 18 | IEPIF | IN endpoint interrupt flag<br>Set by USBHS when one of the IN endpoints in device mode has raised an interrupt. Software should first read USBHS_DAEPINT register to get the device number, and then read the corresponding USBHS_DIEPxINTF register to get the flags of the endpoint that cause the interrupt. This bit will be automatically cleared after the respective endpoint's flags which cause this interrupt are cleared.<br>**Note:** Only accessible in device mode. |
| 17:16 | Reserved | Must be kept at reset value. |
| 15 | EOPFIF | End of periodic frame interrupt flag<br>When USB bus time in a frame has reaches the value defined by EOPFT[1:0] bits in USBHS_DCFG register, USBHS sets this flag.<br>**Note:** Only accessible in device mode. |
| 14 | ISOOPDIF | Isochronous OUT packet dropped interrupt flag<br>USBHS set this bit if it receives an isochronous OUT packet but cannot save it into Rx FIFO because the FIFO doesn't have enough space.<br>**Note:** Only accessible in device mode. |
| 13 | ENUMF | Enumeration finished<br>USBHS sets this bit after the speed enumeration finishes. Software is able to read USBHS_DSTAT register to get the current device speed.<br>**Note:** Only accessible in device mode. |
| 12 | RST | USB reset<br>USBHS sets this bit when it detects a USB reset signal on bus.<br>**Note:** Only accessible in device mode. |
| 11 | SP | USB suspend<br>USBHS sets this bit when it detects that the USB bus is idle for 3 ms and enters suspend state.<br>**Note:** Only accessible in device mode. |
| 10 | ESP | Early suspend<br>USBHS sets this bit when it detects that the USB bus is idle for 3 ms.<br>**Note:** Only accessible in device mode. |
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | GONAK | Global OUT NAK effective<br>Software is able to write 1 to SGONAK bit in the USBHS_DCTL register and USBHS will set GONAK flag after writing to SGONAK takes effect. And this bit can be cleared by writting 1 to CGONAK bit in the USBHS_DCTL register.<br>**Note:** Only accessible in device mode. |
| 6 | GNPINAK | Global IN Non-Periodic NAK effective<br>Software is able to write 1 to SGINAK bit in the USBHS_DCTL register and USBHS |

| | | |
|---|---|---|
| | | will set GNPINAK flag after writing to SGINAK takes effect. And this bit can be cleared by writting 1 to CGINAK bit in the USBHS_DCTL register.<br>**Note:** Only accessible in device mode. |
| 5 | NPTXFEIF | Non-Periodic Tx FIFO empty interrupt flag<br>This interrupt is triggered when the non-periodic transmit FIFO is either half or completely empty. The threshold is determined by the non-periodic Tx FIFO empty level bit (TXFTH) in the USBHS_GAHBCS register.<br>**Note:** Only accessible in host mode. |
| 4 | RXFNEIF | Rx FIFO non-empty interrupt flag<br>USBHS sets this bit when there is at least one packet or status entry in the Rx FIFO.<br>**Note:** Accessible in both host and device modes. |
| 3 | SOF | Start of frame<br>Host Mode: USBHS sets this bit when it prepares to transmit a SOF or Keep-Alive on USB bus. Software can clear this bit by writing 1.<br>Device Mode: USBHS sets this bit to after it receives a SOF token. The application can read the Device Status register to get the current frame number. Software can clear this bit by writing 1.<br>**Note:** Accessible in both host and device modes. |
| 2 | OTGIF | OTG interrupt flag<br>USBHS sets this bit when the flags in USBHS_GOTGINTF register generate a interrupt. Software should read USBHS_GOTGINTF register to get the source of this interrupt. This bit is cleared after the flags in USBHS_GOTGINTF causing this interrupt are cleared.<br>**Note:** Accessible in both host and device modes. |
| 1 | MFIF | Mode fault interrupt flag<br>USBHS sets this bit if software operates host-only register in device mode, or operates device-mode in host mode. These fault operations won't take effect.<br>**Note:** Accessible in both host and device modes. |
| 0 | COPM | Current operation mode<br>0: Device mode<br>1: Host mode<br>**Note:** Accessible in both host and device modes. |

### Global interrupt enable register (USBHS_GINTEN)

Address offset: 0x0018
Reset value: 0x0000 0000

This register works with the global interrupt flag register (USBHS_GINTF) to interrupt the application. When an interrupt enable bit is disabled, the interrupt associated with that bit is not generated. However, the global Interrupt flag register bit corresponding to that interrupt is still set.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WKUPIE | SESIE | DISCIE | IDPSCIE | LPMIE | PTXFEIE | HCIE | HPIE | Reserved | | PXNCIE/ ISOONCIE | ISOINCIE | OEPIE | IEPIE | Reserved | |
| rw | rw | rw | rw | rw | rw | rw | r | | | rw | rw | rw | rw | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EOPFIE | ISOOPDIE | ENUMFIE | RSTIE | SPIE | ESPIE | Reserved | | GONAKIE | GNPINAKIE | NPTXFEIE | RXFNEIE | SOFIE | OTGIE | MFIE | Reserved |
| rw | rw | rw | rw | rw | rw | | | rw | rw | rw | rw | rw | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | WKUPIE | Wakeup interrupt enable<br>0: Disable wakeup interrupt<br>1: Enable wakeup interrupt<br>**Note:** Accessible in both host and device modes. |
| 30 | SESIE | Session interrupt enable<br>0: Disable session interrupt<br>1: Enable session interrupt<br>**Note:** Accessible in both host and device modes. |
| 29 | DISCIE | Disconnect interrupt enable<br>0: Disable disconnect interrupt<br>1: Enable disconnect interrupt<br>**Note:** Only accessible in device mode. |
| 28 | IDPSCIE | ID pin status change interrupt enable<br>0: Disable connector ID pin status interrupt<br>1: Enable connector ID pin status interrupt<br>**Note:** Accessible in both host and device modes. |
| 27 | LPMIE | LPM interrupt enable<br>0: disable LPM interrupt<br>1: enable LPM interrupt<br>**Note:** Accessible in both host and device modes. |
| 26 | PTXFEIE | Periodic Tx FIFO empty interrupt enable<br>0: Disable periodic Tx FIFO empty interrupt<br>1: Enable periodic Tx FIFO empty interrupt<br>**Note:** Only accessible in host mode. |
| 25 | HCIE | Host channels interrupt enable<br>0: Disable host channels interrupt<br>1: Enable host channels interrupt |

| | | |
|---|---|---|
| | | **Note:** Only accessible in host mode. |
| 24 | HPIE | Host port interrupt enable<br>0: Disable host port interrupt<br>1: Enable host port interrupt<br>**Note:** Only accessible in host mode. |
| 23:22 | Reserved | Must be kept at reset value. |
| 21 | PXNCIE | Periodic transfer not complete interrupt enable<br>0: Disable Periodic transfer not complete interrupt<br>1: Enable Periodic transfer not complete interrupt<br>**Note:** Only accessible in host mode. |
| | ISOONCIE | Isochronous OUT transfer not complete interrupt enable<br>0: Disable Isochronous OUT transfer not complete interrupt<br>1: Enable Isochronous OUT transfer not complete interrupt<br>**Note:** Only accessible in device mode. |
| 20 | ISOINCIE | Isochronous IN transfer not complete interrupt enable<br>0: Disable Isochronous IN transfer not complete interrupt<br>1: Enable Isochronous IN transfer not complete interrupt<br>**Note:** Only accessible in device mode. |
| 19 | OEPIE | OUT endpoints interrupt enable<br>0: Disable OUT endpoints interrupt<br>1: Enable OUT endpoints interrupt<br>**Note:** Only accessible in device mode. |
| 18 | IEPIE | IN endpoints interrupt enable<br>0: Disable IN endpoints interrupt<br>1: Enable IN endpoints interrupt<br>**Note:** Only accessible in device mode. |
| 17:16 | Reserved | Must be kept at reset value. |
| 15 | EOPFIE | End of periodic frame interrupt enable<br>0: Disable end of periodic frame interrupt<br>1: Enable end of periodic frame interrupt<br>**Note:** Only accessible in device mode. |
| 14 | ISOOPDIE | Isochronous OUT packet dropped interrupt enable<br>0: Disable isochronous OUT packet dropped interrupt<br>1: Enable isochronous OUT packet dropped interrupt<br>**Note:** Only accessible in device mode. |
| 13 | ENUMFIE | Enumeration finish enable<br>0: Disable enumeration finish interrupt<br>1: Enable enumeration finish interrupt<br>**Note:** Only accessible in device mode. |

| | | |
|---|---|---|
| 12 | RSTIE | USB reset interrupt enable<br>0: Disable USB reset interrupt<br>1: Enable USB reset interrupt<br>**Note:** Only accessible in device mode. |
| 11 | SPIE | USB suspend interrupt enable<br>0: Disable USB suspend interrupt<br>1: Enable USB suspend interrupt<br>**Note:** Only accessible in device mode. |
| 10 | ESPIE | Early suspend interrupt enable<br>0: Disable early suspend interrupt<br>1: Enable early suspend interrupt<br>**Note:** Only accessible in device mode. |
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | GONAKIE | Global OUT NAK effective interrupt enable<br>0: Disable global OUT NAK interrupt<br>1: Enable global OUT NAK interrupt<br>**Note:** Only accessible in device mode. |
| 6 | GNPINAKIE | Global non-periodic IN NAK effective interrupt enable<br>0: Disable global non-periodic IN NAK effective interrupt<br>1: Enable global non-periodic IN NAK effective interrupt<br>**Note:** Only accessible in device mode. |
| 5 | NPTXFEIE | Non-periodic Tx FIFO empty interrupt enable<br>0: Disable non-periodic Tx FIFO empty interrupt<br>1: Enable non-periodic Tx FIFO empty interrupt<br>**Note:** Only accessible in Host mode. |
| 4 | RXFNEIE | Receive FIFO non-empty interrupt enable<br>0: Disable receive FIFO non-empty interrupt<br>1: Enable receive FIFO non-empty interrupt<br>**Note:** Accessible in both device and host modes. |
| 3 | SOFIE | Start of frame interrupt enable<br>0: Disable start of frame interrupt<br>1: Enable start of frame interrupt<br>**Note:** Accessible in both device and host modes. |
| 2 | OTGIE | OTG interrupt enable<br>0: Disable OTG interrupt<br>1: Enable OTG interrupt<br>**Note:** Accessible in both device and host modes. |
| 1 | MFIE | Mode fault interrupt enable<br>0: Disable mode fault interrupt |

| | | |
|---|---|---|
| | | 1: Enable mode fault interrupt<br>**Note:** Accessible in both device and host modes. |
| 0 | Reserved | Must be kept at reset value. |

### Global receive status read/receive status read and pop registers (USBHS_GRSTATR/USBHS_GRSTATP)

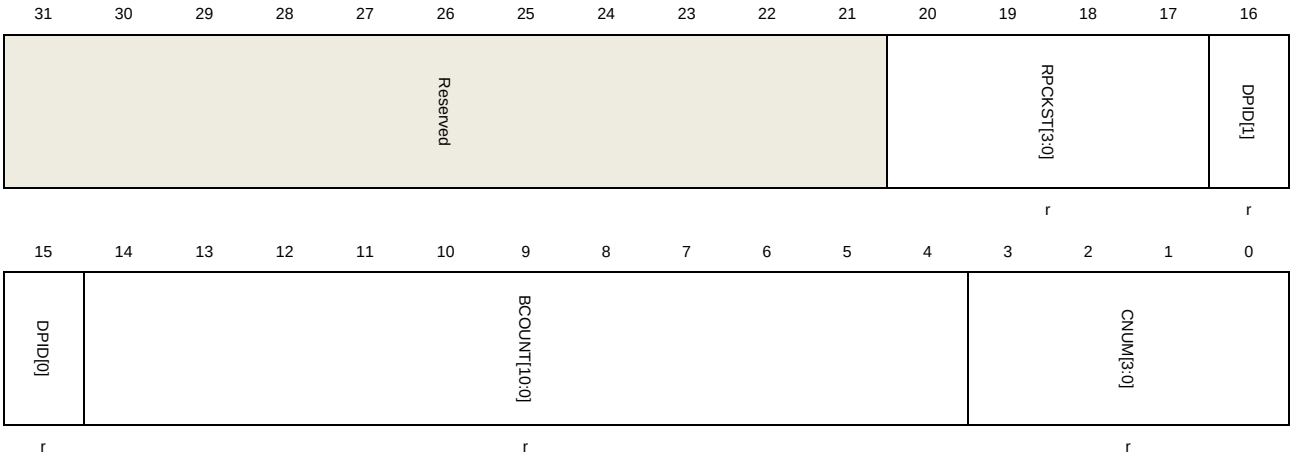
Address offset for Read: 0x001C
Address offset for Pop: 0x0020
Reset value: 0x0000 0000

A read to the receive status read register returns the entry of the top of the Rx FIFO. A read to the Receive status read and pop register additionally pops the top entry out of the Rx FIFO.

The entries in RxFIFO have different meanings in host and device modes. Software should only read this register after when Receive FIFO non-empty interrupt flag bit of the global interrupt flag register (RXFNEIF bit in USBHS_GINTF) is triggered.

This register has to be accessed by word (32-bit)

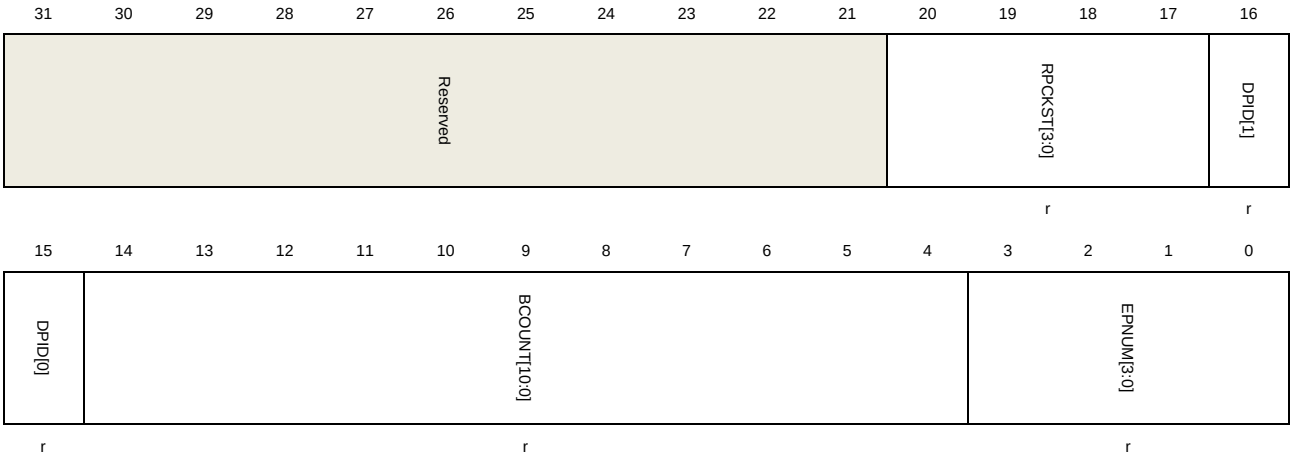
**Host mode:**




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:17 | RPCKST[3:0] | Received packet status<br>0010: IN data packet received<br>0011: IN transfer completed (generates an interrupt if poped)<br>0101: Data toggle error (generates an interrupt if poped)<br>0111: Channel halted (generates an interrupt if poped)<br>Others: Reserved |
| 16:15 | DPID[1:0] | Data PID<br>The Data PID of the received packet |

| | | |
|---|---|---|
| | | 00: DATA0<br>10: DATA1<br>01: DATA2<br>11: MDATA |
| 14:4 | BCOUNT[10:0] | Byte count<br>The byte count of the received IN data packet. |
| 3:0 | CNUM[3:0] | Channel number<br>The channel number to which the current received packet belongs. |

**Device mode:**




| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:17 | RPCKST[3:0] | Received packet status<br>0001: Global OUT NAK (generates an interrupt)<br>0010: OUT data packet received<br>0011: OUT transfer completed (generates an interrupt)<br>0100: SETUP transaction completed (generates an interrupt)<br>0110: SETUP data packet received<br>Others: Reserved |
| 16:15 | DPID[1:0] | Data PID<br>The Data PID of the received OUT data packet<br>00: DATA0<br>10: DATA1<br>01: DATA2<br>11: MDATA |
| 14:4 | BCOUNT[10:0] | Byte count<br>The byte count of the received data packet. |
| 3:0 | EPNUM[3:0] | Endpoint number |

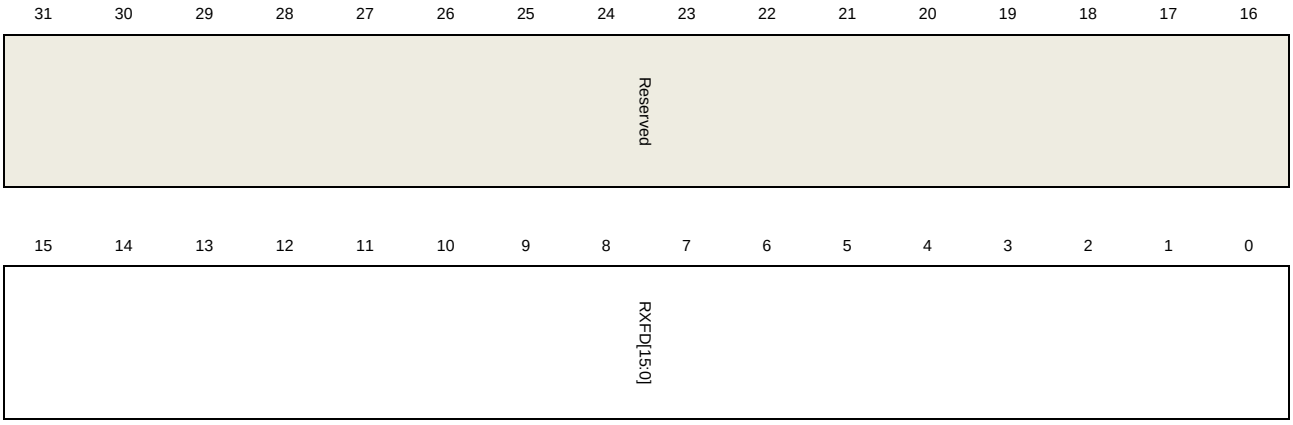
The endpoint number to which the current received packet belongs.

### Global receive FIFO length register (USBHS_GRFLEN)

Address offset: 0x024

Reset value: 0x0000 0200

This register has to be accessed by word (32-bit)



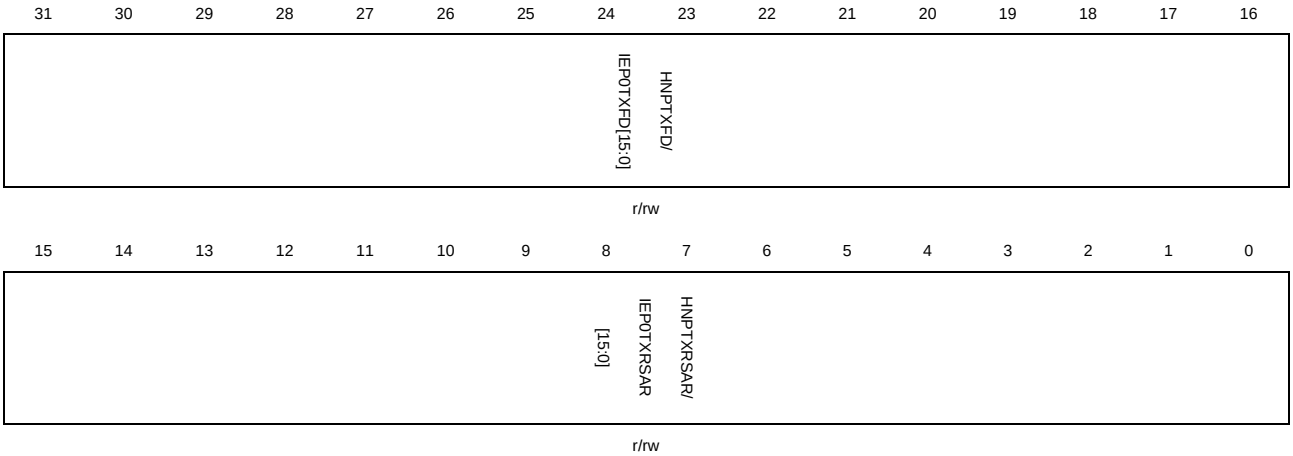

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | RXFD[15:0] | Rx FIFO depth<br>In terms of 32-bit word.<br>1≤RXFD≤1024 |

### Host non-periodic transmit FIFO length register /Device IN endpoint 0 transmit FIFO length (USBHS_HNPTFLEN/USBHS_DIEP0TFLEN)

Address offset: 0x028

Reset value: 0x0200 0200

This register has to be accessed by word (32-bit)



**Host Mode:**

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | HNPTXFD[15:0] | Non-periodic Tx FIFO depth<br>In terms of 32-bit word.<br>1≤HNPTXFD≤1024 |
| 15:0 | HNPTXRSAR[15:0] | Non-periodic Tx RAM start address<br>The start address for non-periodic transmit FIFO RAM in terms of 32-bit word. |

**Device Mode:**

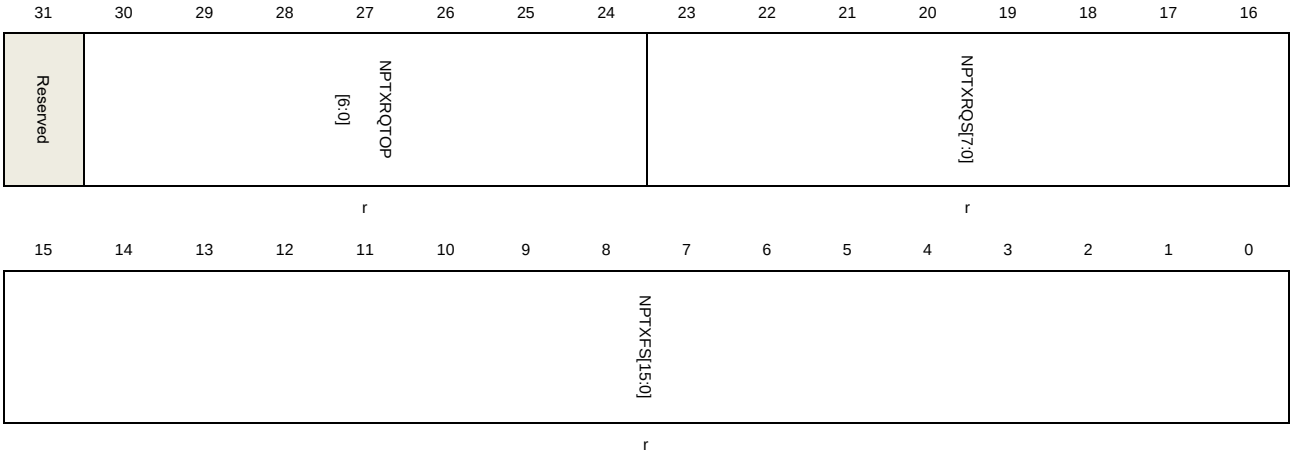
| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | IEP0TXFD[15:0] | IN Endpoint 0 Tx FIFO depth<br>In terms of 32-bit words.<br>16≤IEP0TXFD≤140 |
| 15:0 | IEP0TXRSAR[15:0] | IN Endpoint 0 TX RAM start address<br>The start address for endpoint0 transmit FIFO RAM in terms of 32-bit word. |

### Host non-periodic transmit FIFO/queue status register (USBHS_HNPTFQSTAT)

Address offset: 0x002C
Reset value: 0x0008 0200

This register has to be accessed by word (32-bit)

This register reports the current status of the non-periodic Tx FIFO and request queue. The request queue holds IN, OUT or other request entries in host mode.
Note: In Device mode, this register is not valid.




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:24 | NPTXRQTOP[6:0] | Top entry of the non-periodic Tx request queue<br>Entry in the non-periodic transmit request queue. |

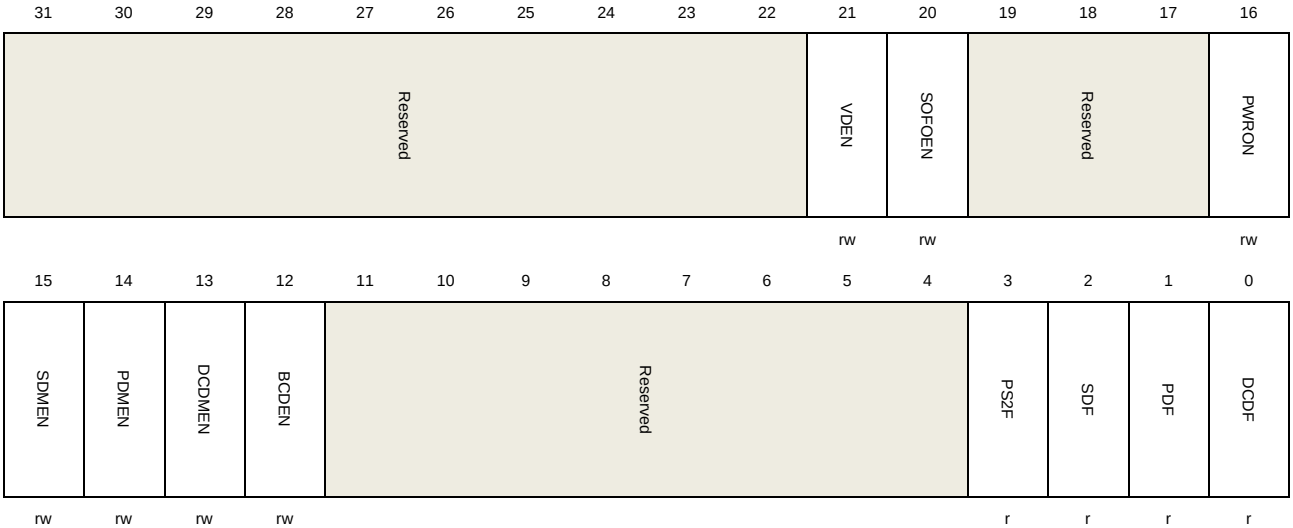
Bits 30:27: Channel number
Bits 26:25:
– 00: IN/OUT token
– 01: Zero-length OUT packet
– 11: Channel halt request
Bit 24: Terminate Flag, indicating last entry for selected channel.

| | | |
|---|---|---|
| 23:16 | NPTXRQS[7:0] | Non-periodic Tx request queue space<br>The remaining space of the non-periodic transmit request queue.<br>0: Request queue is Full<br>1: 1 entry<br>2: 2 entries<br>…<br>n: n entries (0≤n≤8)<br>Others: Reserved |
| 15:0 | NPTXFS[15:0] | Non-periodic Tx FIFO space<br>The remaining space of the non-periodic transmit FIFO.<br>In terms of 32-bit words.<br>0: Non-periodic Tx FIFO is full<br>1: 1 words<br>2: 2 words<br>…<br>n: n words (0≤n≤NPTXFD)<br>Others: Reserved |

### Global core configuration register (USBHS_GCCFG)

Address offset: 0x0038
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)



| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21 | VDEN | Enable of VBUS sensing comparator to detect VBUS valid. VBUS comparator is enabled automatically if HNP or SRP is supported.<br>0: VBUS detection is disabled<br>1: VBUS detection is enabled |
| 20 | SOFOEN | SOF output enable<br>0: SOF pulse output disabled.<br>1: SOF pulse output enabled. |
| 19:17 | Reserved | Must be kept at reset value. |
| 16 | PWRON | Power on<br>This bit is the power switch for the internal embedded PHY.<br>0: Embedded PHY power off.<br>1: Embedded PHY power on. |
| 15 | SDMEN | Secondary detection mode enable<br>0: Secondary detection disable<br>1: Secondary detection enable |
| 14 | PDMEN | Primary detection mode enable<br>0: Primary detection disable<br>1: Primary detection enable |
| 13 | DCDMEN | Data connect detection mode enable<br>0: Data connect detection disable<br>1: Data connect detection enable |
| 12 | BCDEN | Battery charging detection enable<br>0: Battery charging detection disable<br>1: Battery charging detection enable |
| 11:4 | Reserved | Must be kept at reset value. |
| 3 | PS2F | PS2 detection status, it is active only in Primary detection mode<br>0: Normal port is detected<br>1: PS2 port is detected |
| 2 | SDF | Secondary detection status<br>0: CDP is detected<br>1: DCP is detected |
| 1 | PDF | Primary detection status<br>0: no BCD supported is detected<br>1: BCD supported is detected |
| 0 | DCDF | Data connect detection status<br>0: Data line connect is not detected |

1: Data line connect is detected

### Core ID register (USBHS_CID)

Address offset: 0x003C

Reset value: 0x0000 1000

This register contains the Product ID.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CID[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CID[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CID[31:0] | Core ID<br>Software can write or read this field and uses this field as a unique ID for its application. |

### Global core LPM configuration register (USBHS_GLPMCFG)

Address offset: 0x0054

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | BESLEN | LPMRCS[2:0] | | | LPMSND | LPMRC[2:0] | | | LPMCHI[3:0] | | | | RSOK |
| | | | rw | r | | | rs | rw | | | rw | | | | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LPMSLPS | LPMRSP[1:0] | | DSEN | BESLTH[3:0] | | | | SSEN | REW | BESL[3:0] | | | | ACKLPM | LPMEN |
| r | r | | rw | rw | | | | rw | rw/r | rw/r | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28 | BESLEN | LPM Errata selection enable<br>0: USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification is selected<br>1: Errata for USB 2.0 ECN: Link Power Management (LPM) is selected |
| 27:25 | LPMRCS[2:0] | LPM retry count status<br>**Note:** Only accessible in host mode |
| 24 | LPMSND | Send LPM transaction<br>When ACK, STALL or NYET response is received, or all the LPM of retry count have been sent, the hardware clears this bit.<br>**Note:** Only accessible in host mode |
| 23:21 | LPMRC[2:0] | LPM retry count<br>It is the number of retry count when an ERROR response is received, until ACK, STALL or NYET response is received<br>**Note:** Only accessible in host mode |
| 20:17 | LPMCHI[3:0] | Channel number index when send LPM transaction<br>**Note:** Only accessible in host mode |
| 16 | RSOK | Resume can be sent after sleep state<br>Host or device can send resume from sleep state after 50us (TL1Residency).<br>When LPMSLPS is 0, it is reset.<br>1: Resume can be started from sleep state<br>0: Resume cannot be started from sleep state |
| 15 | LPMSLPS | Sleep status<br>**Host mode:**<br>The host transitions to sleep status after receiving ACK response.<br>**Device mode:**<br>The device enters into sleep status after sending ACK response and the TL1TokenRetry timer has been expired.<br>1: Core is in sleep status<br>0: Core is not in sleep status |
| 14:13 | LPMRSP[1:0] | Response of LPM<br>11: ACK<br>10: NYET<br>01: STALL<br>00: ERROR (no response) |
| 12 | DSEN | Deep sleep enable<br>Enable suspending the PHY in deep sleep mode |

| | | |
|---|---|---|
| 11:8 | BESLTH[3:0] | BESL threshold<br>**Device mode:**<br>When BESL is greater than or equal to the BESLTH value, device enters into deep low power mode.<br>**Host mode:**<br>BESLTH indicates the duration of resume signal (TL1HubDrvResume2) when it detects device initialed resume.<br>0000: 75us<br>0001: 100us<br>0010: 150us<br>0011: 250us<br>0101: 450us<br>0110: 950us |
| 7 | SSEN | Shallow sleep enable<br>Enable suspending the PHY in shallow sleep mode |
| 6 | REW | bRemoteWake value<br>**Host mode:**<br>The remote wake up value to be sent in LPM transaction<br>**Device mode (read-only):**<br>When ACK, STALL or NYET is sent, it is updated with bRemoteWake value in received LPM transaction |
| 5:2 | BESL[3:0] | Best effort service latency<br>**Host mode:**<br>BESL value to be sent in LPM transaction. It is also the duration of resume (TL1HubDrvResume1) when host initialed resume.<br>**Device mode:**<br>When ACK, STALL or NYET is sent, it is updated with BESL value in received LPM transaction<br>0000: 125us<br>0001: 150us<br>0010: 200us<br>0011: 300us<br>0100: 400us<br>0101: 500us<br>0110: 1000us<br>0111: 2000us<br>1000: 3000us<br>1001: 4000us<br>1010: 5000us<br>1011: 6000us<br>1100: 7000us<br>1101: 8000us<br>1110: 9000us |

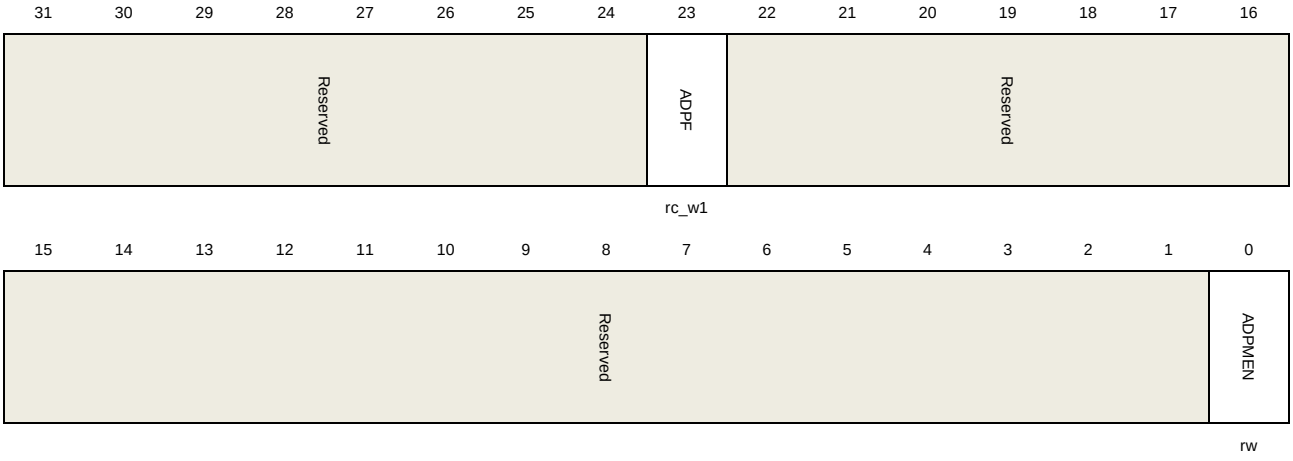
| | | |
|---|---|---|
| | | 1111: 10000us |
| 1 | ACKLPM | ACK in LPM transaction enable<br>1: ACK<br>The device response with ACK only on successful LPM transaction.<br>- No ERROR in LPM transaction<br>- No data pending error<br>- bLinkState = 0001 in received LPM transaction<br>0: NYET<br>The device response with NYET<br>- The received bLinkState value is not 0001<br>- There is an error in received LPM transaction<br>**Note:** Only accessible in device mode |
| 0 | LPMEN | LPM enable<br>1: Enable LPM<br>0: Disable LPM |

### Power down register (USBHS_PWRD)

Address offset: 0x0058

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | ADPF | ADP event interrupt flag |
| 22:1 | Reserved | Must be kept at reset value. |
| 0 | ADPMEN | ADP module enable<br>1: ADP module is enable<br>0: ADP module is disable |

### ADP control and status register (USBHS_ADPCTL)

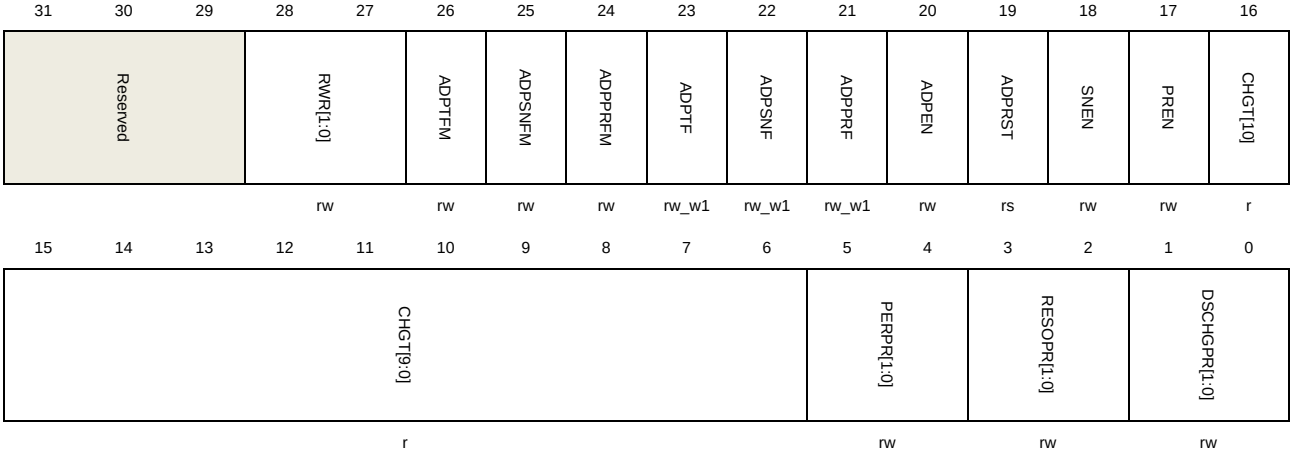
Address offset: 0x0060

Reset value: 0x0000 0000

In order to write in the register, program RWR with 10 and keep polling until RWR = 00.

In order to read from the register, wait any ADP flag is set or write RWR with 01 and keep polling until RWR = 00.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:29 | Reserved | Must be kept at reset value. |
| 28:27 | RWR[1:0] | Read and write request<br>00: Read or write valid (updated by core)<br>01: Read request<br>10: Write request |
| 26 | ADPTFM | The mask of ADP timeout interrupt flag |
| 25 | ADPSNFM | The mask of ADP sense interrupt flag |
| 24 | ADPPRFM | The mask of ADP probe interrupt flag |
| 23 | ADPTF | ADP timeout interrupt flag |
| 22 | ADPSNF | ADP sense interrupt flag |
| 21 | ADPPRF | ADP probe interrupt flag |
| 20 | ADPEN | ADP enable<br>1: ADP is enable<br>0: ADP is disable |
| 19 | ADPRST | ADP reset<br>This is cleared automatically by core after reset procedure |
| 18 | SNEN | ADP sense enable |

| | | |
|---|---|---|
| | | 1: Sense is enable<br>0: Sense is disable |
| 17 | PREN | ADP probe enable<br>1: Probe is enable<br>0: Probe is disable |
| 16:6 | CHGT[10:0] | The latest time that VBUS ramps from VADPSINK to VADPPRB. These bits are defined in units of 32 kHz clock cycle.<br>000: 1 cycle<br>001: 2 cycles<br>002: 3 cycles<br>003: 4 cycles<br>…<br>7ff: 2048 cycles |
| 5:4 | PERPR[1:0] | Period of probe<br>00: 0.625 to 0.925 second<br>01: 1.25 to 1.85 second<br>10: 1.9 to 2.6 second |
| 3:2 | RESOPR[1:0] | The resolution of CHGT value. These bits are defined in units of 32 kHz clock cycle. If 10 is chosen, the CHGT increments for every three 32 kHz clock cycle.<br>00: 1 cycle<br>01: 2 cycles<br>10: 3 cycles<br>11: 4 cycles |
| 1:0 | DSCHGPR[1:0] | Time of probe discharge<br>00: 4 ms<br>01: 8 ms<br>10: 16 ms<br>11: 32 ms |

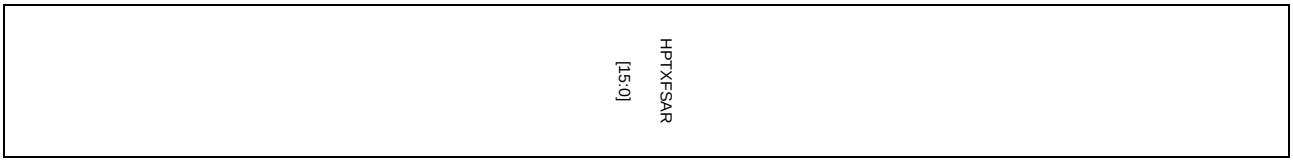
### Host periodic transmit FIFO length register (USBHS_HPTFLEN)

Address offset: 0x0100

Reset value: 0x0200 0600

This register has to be accessed by word (32-bit)



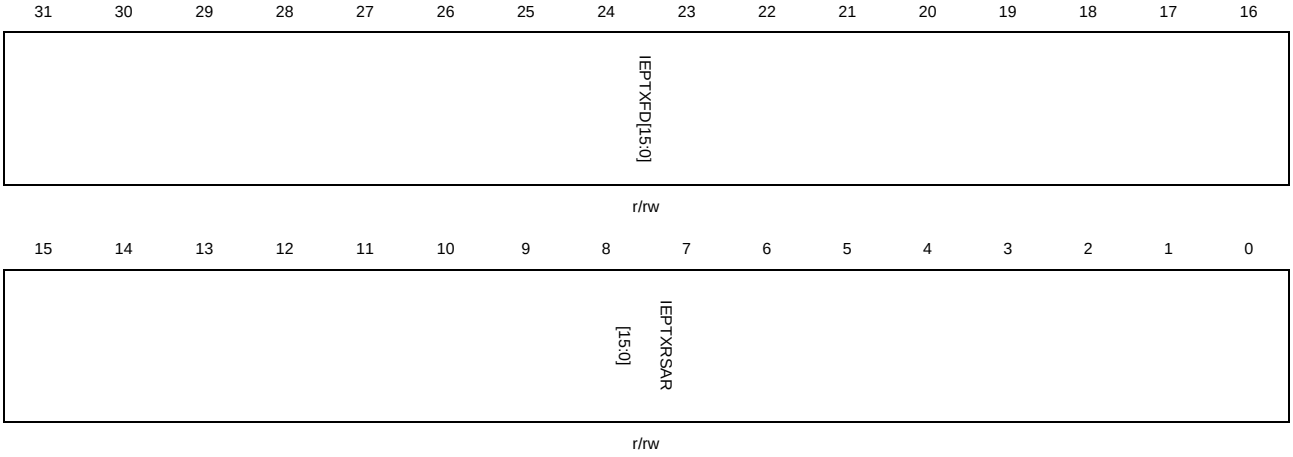



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | HPTXFD[15:0] | Host Periodic Tx FIFO depth<br>In terms of 32-bit word.<br>1≤HPTXFD≤1024 |
| 15:0 | HPTXFSAR[15:0] | Host periodic Tx RAM start address<br>The start address for host periodic transmit FIFO RAM in terms of 32-bit word. |

### Device IN endpoint transmit FIFO length register (USBHS_DIEPxTFLEN) (x = 1..7, where x is the FIFO_number)

Address offset: 0x0104 + (FIFO_number – 1) × 0x04
Reset value: 0x0200 0400

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | IEPTXFD[15:0] | IN endpoint Tx FIFO x depth<br>In terms of 32-bit word.<br>1≤IEPTXFD≤1024 |
| 15:0 | IEPTXRSAR[15:0] | IN endpoint FIFOx Tx x RAM start address<br>The start address for IN endpoint transmit FIFO x in terms of 32-bit word. |

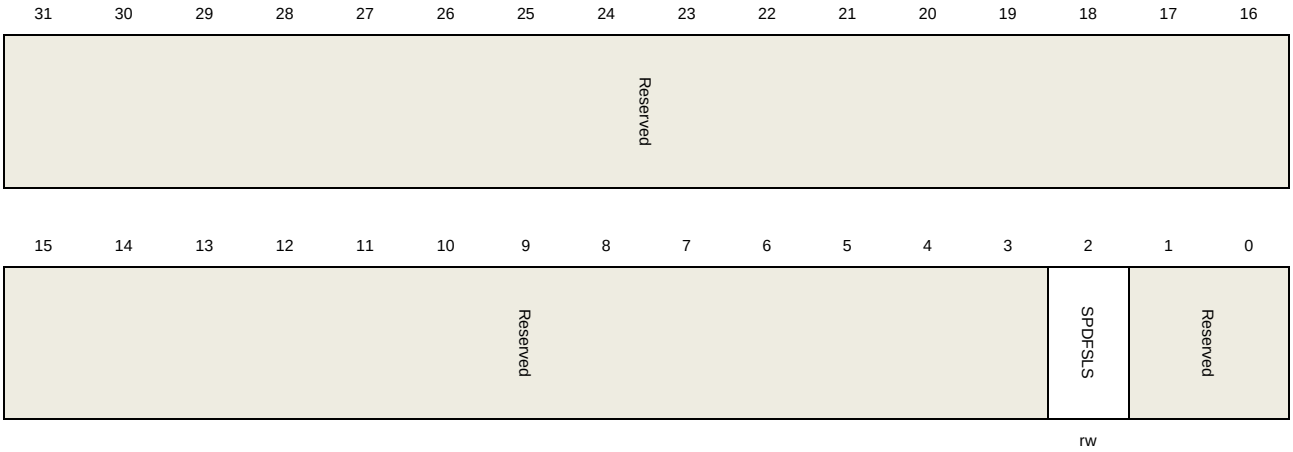
## 49.7.2. Host control and status registers

### Host control register (USBHS_HCTL)

Address offset: 0x0400
Reset value: 0x0000 0000

This register configures the core after power-on in host mode. Do not modify it after host initialization.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | SPDFSLS | Speed limited to FS and LS<br>Software may use this bit to limit USBHS's enumeration speed to FS/LS and make USBHS not perform High-Speed enumeration during reset.<br>0: Speed not limited.<br>1: Speed limited in FS/LS only. |
| 1:0 | Reserved | Must be kept at reset value. |

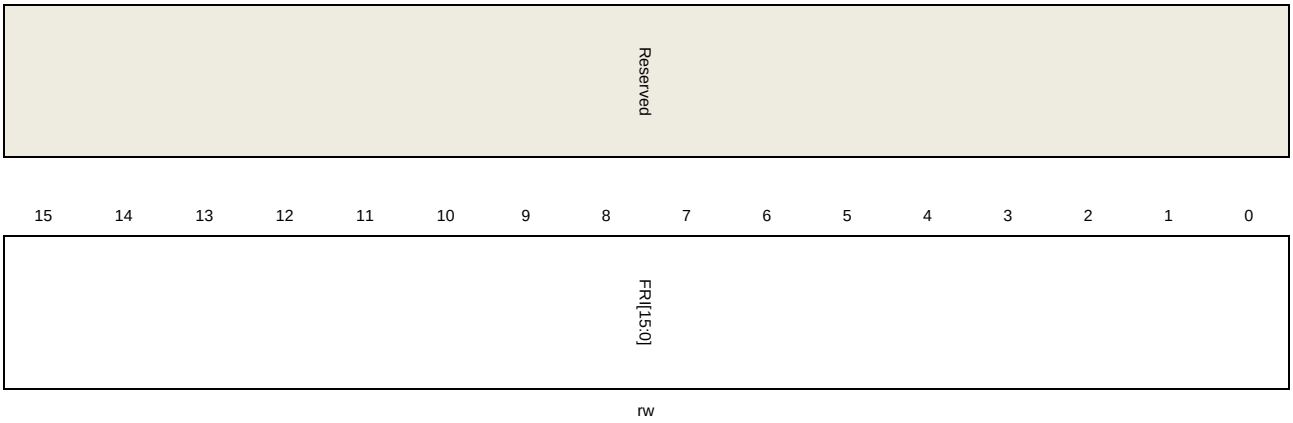
### Host frame interval register (USBHS_HFT)

Address offset: 0x0404
Reset value: 0x0000 EA60

This register sets the frame interval for the current enumerating speed when USBHS controller is enumerating.

This register has to be accessed by word (32-bit)





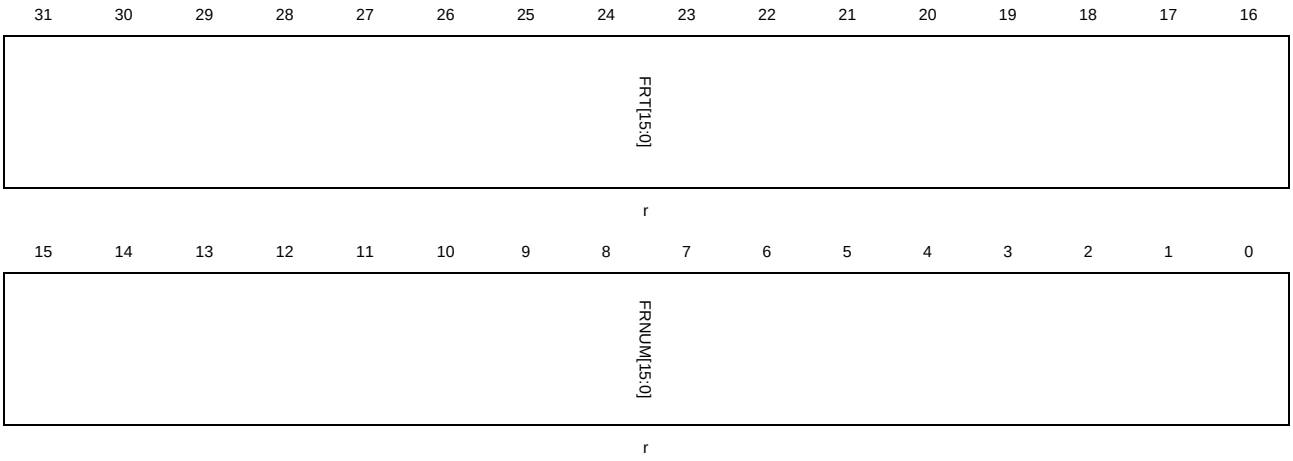

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | FRI[15:0] | Frame interval<br>This value describes the frame time in terms of PHY clock. Port is enabled after a port reset operation, USBHS uses a proper value according to the current speed, and software can write to this field to change the value. This value should be calculated using the frequency described below:<br>Internal Embedded PHY:<br>High-Speed: 60MHz<br>Full-Speed / Low-Speed: 48MHz<br>External ULPI PHY: 60MHz<br>For the GD32H75E, low speed: 6 MHz. |

### Host frame information remaining register (USBHS_HFINFR)

Address offset: 0x408

Reset value: 0xEA60 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | FRT[15:0] | Frame remaining time |

| | | |
|---|---|---|
| | | This field reports the remaining time of current frame in terms of PHY clock. |
| 15:0 | FRNUM[15:0] | Frame number<br>This field reports the frame number of current frame and returns to 0 after it reaches 0x3FF. |

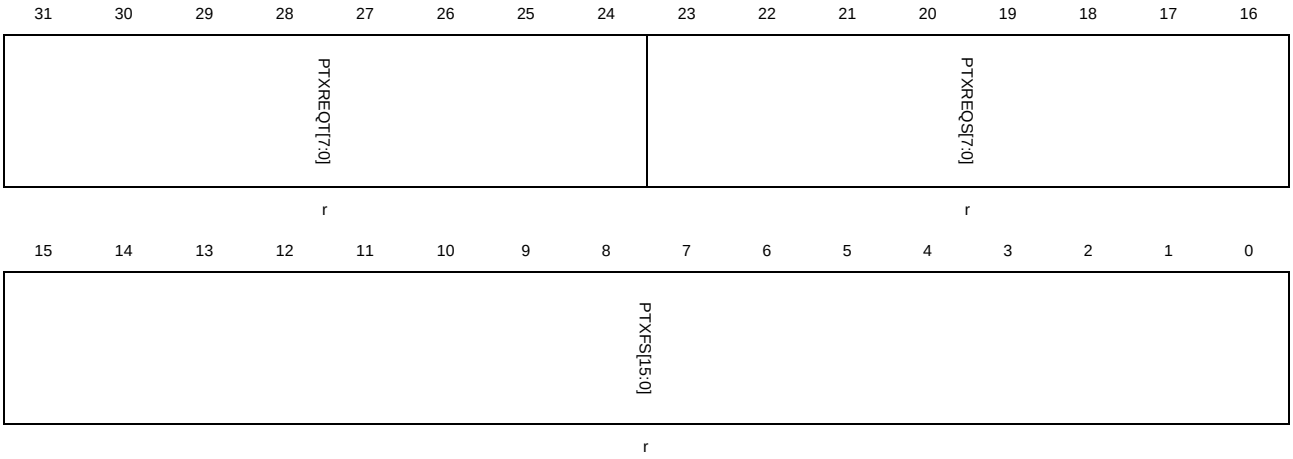
### Host periodic transmit FIFO/queue status register (USBHS_HPTFQSTAT)

Address offset: 0x0410

Reset value: 0x0008 0200

This register reports the current status of the host periodic Tx FIFO and request queue. The request queue holds IN, OUT or other request entries in host mode.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | PTXREQT[7:0] | Top entry of the periodic Tx request queue<br>Entry in the periodic transmit request queue.<br>Bit 31: Odd/Even frame<br>– 0: send in even frame<br>– 1: send in odd frame<br>Bits 30:27: Channel Number<br>Bits 26:25:<br>– 00: IN/OUT token<br>– 01: Zero-length OUT packet<br>– 11: Channel halt request<br>Bit 24: Terminate Flag, indicating last entry for selected channel. |
| 23:16 | PTXREQS[7:0] | Periodic Tx request queue space<br>The remaining space of the periodic transmit request queue.<br>0: Request queue is Full<br>1: 1 entry<br>2: 2 entries<br>… |

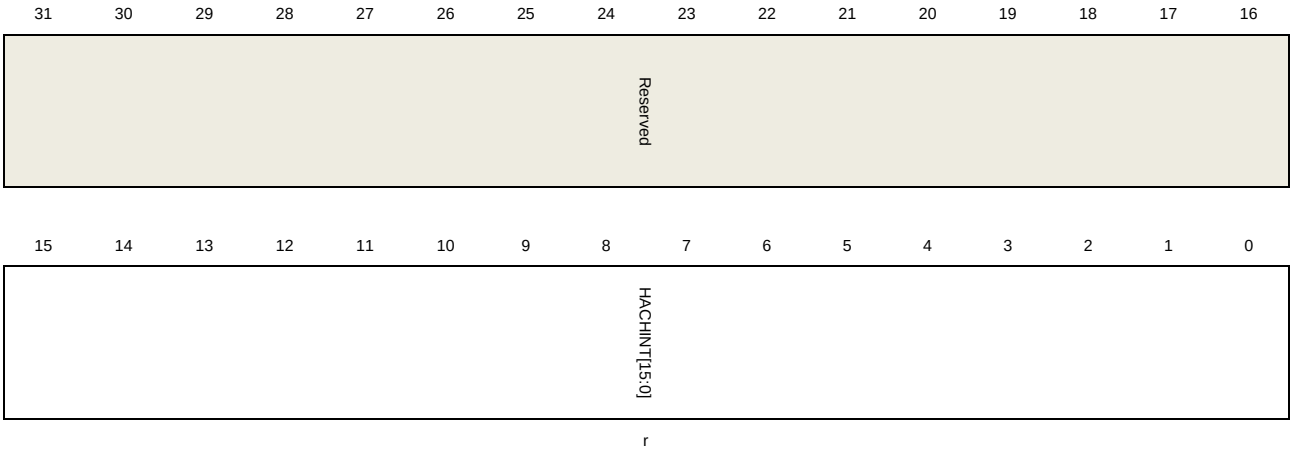
| | | |
|---|---|---|
| | | n: n entries (0≤n≤8)<br>Others: Reserved |
| 15:0 | PTXFS[15:0] | Periodic Tx FIFO space<br>The remaining space of the periodic transmit FIFO.<br>In terms of 32-bit word.<br>0: periodic Tx FIFO is full<br>1: 1 word<br>2: 2 words<br>…<br>n: n words (0≤n≤PTXFD)<br>Others: Reserved |

### Host all channels interrupt register (USBHS_HACHINT)

Address offset: 0x0414
Reset value: 0x0000 0000

When a channel interrupt is triggered, USBHS sets corresponding bit in this register and software should read this register to know which channel is asserting interrupt.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | HACHINT[15:0] | Host all channel interrupts<br>Each bit represents a channel: Bit 0 for channel 0, bit 15 for channel 15. |

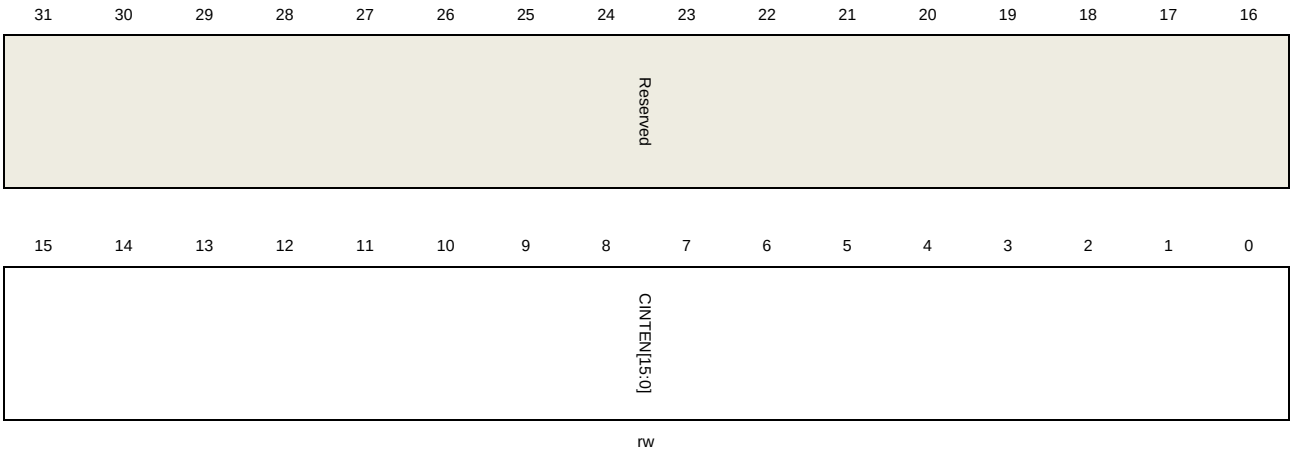
### Host all channels interrupt enable register (USBHS_HACHINTEN)

Address offset: 0x0418
Reset value: 0x0000 0000

This register can be used by software to enable or disable a channel's interrupt. Only the channel whose corresponding bit in this register is set is able to cause the channel interrupt

flag HCIF in USBHS_GINTF register.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | CINTEN[15:0] | Channel interrupt enable<br>0: Disable channel-n interrupt<br>1: Enable channel-n interrupt<br>Each bit represents a channel: Bit 0 for channel 0, bit 15 for channel 15. |

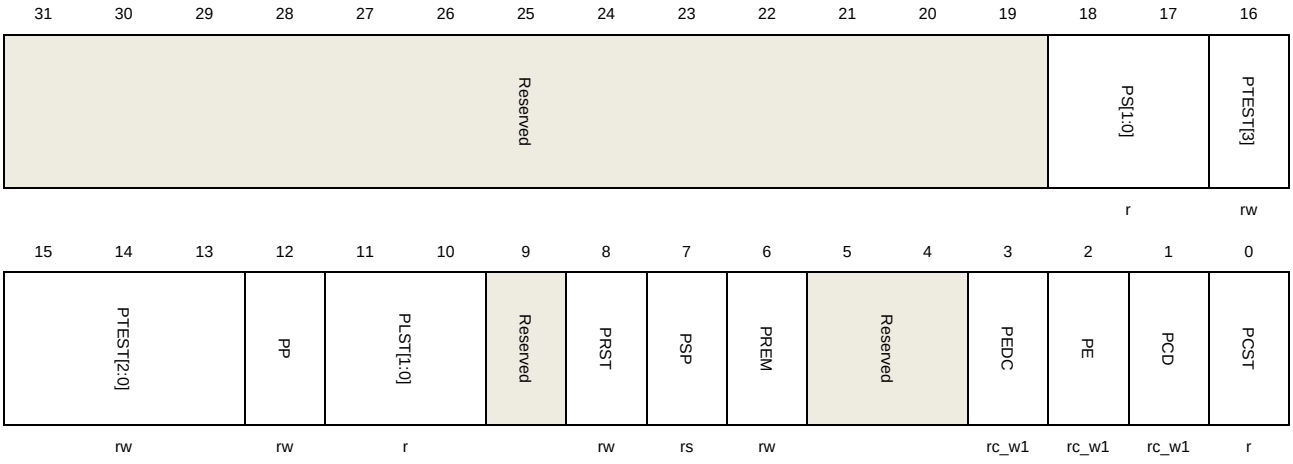
### Host port control and status register (USBHS_HPCS)

Address offset: 0x0440

Reset value: 0x0002 0000

This register controls the port's behavior and also has some flags which report the status of the port. The HPIF flag in USBHS_GINTF register will be triggered if one of these flags in this register is set by USBHS: PRST, PEDC and PCD.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18:17 | PS[1:0] | Port speed<br>Report the enumerated speed of the device attached to this port.<br>00: High-Speed<br>01: Full-Speed<br>10: Low-Speed<br>Others: Reserved |
| 16:13 | PTEST[3:0] | Port Test control<br>The software writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is sended on the port. When test mode is used, The HS_CUR_FE bit in USBHS_GUSBCS register should also be set.<br>0000: Test mode disabled<br>0001: Test_J mode<br>0010: Test_K mode<br>0011: Test_SE0_NAK mode<br>0100: Test_Packet mode<br>0101: Test_Force_Enable<br>Others: Reserved |
| 12 | PP | Port power<br>This bit should be set before a port is used. Because USBHS doesn't have power supply ability, it only uses this bit to know whether the port is in powered state. Software should ensure the true power supply on $V_{BUS}$ before setting this bit.<br>0: Port is powered off<br>1: Port is powered on |
| 11:10 | PLST[1:0] | Port line status<br>Report the current state of USB data lines<br>Bit 10: State of DP line<br>Bit 11: State of DM line |
| 9 | Reserved | Must be kept at reset value. |
| 8 | PRST | Port reset<br>Application sets this bit to start a reset signal on USB port. Application should clear this bit when it wants to stop the reset signal.<br>0: Port is not in reset state<br>1: Port is in reset state |
| 7 | PSP | Port suspend<br>Application sets this bit to put port into suspend state. When this bit is set the port stops sending SOF tokens. This bit can only be cleared by the following operations:<br>– PRST bit in this register is set by application<br>– PREM bit in this register is set |

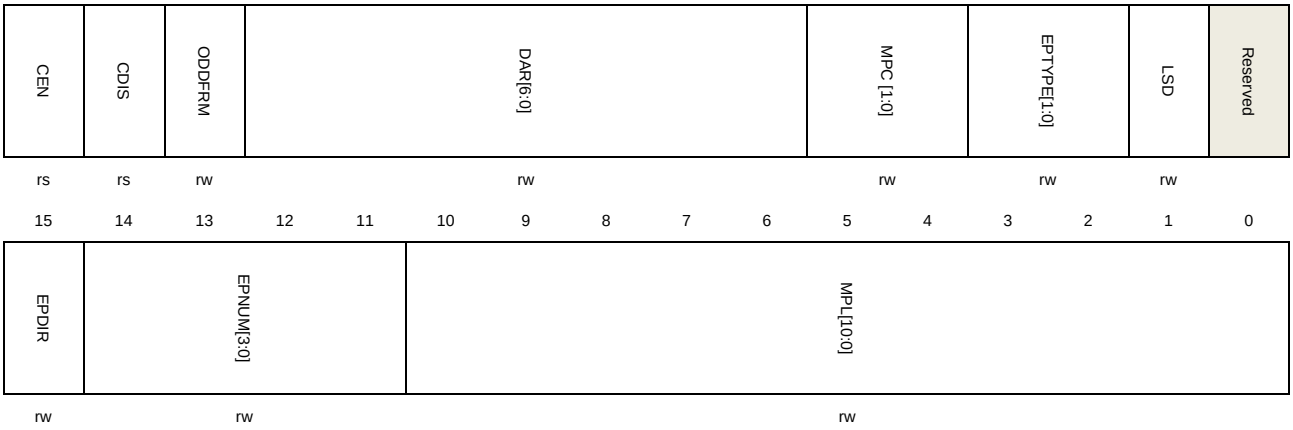
| | | |
|---|---|---|
| | | – A remote wakeup signal is detected<br>– A device disconnection is detected<br>0: Port is not in suspend state<br>1: Port is in suspend state |
| 6 | PREM | Port resume<br>Application sets this bit to start a resume signal on USB port. Application should clear this bit when it wants to stop the resume signal.<br>0: No resume driven<br>1: Resume driven<br>When the application sets PREM in sleep status, the core continues to drive the resume signal until the timer specified with BESLTH has expired.<br>When the core detects a USB remote wakeup, it starts driving the resume signal and clears it automatically at the end of resume. |
| 5:4 | Reserved | Must be kept at reset value. |
| 3 | PEDC | Port enable/disable change<br>Set by the core when the status of the Port enable bit 2 in this register changes. |
| 2 | PE | Port Enable<br>This bit is automatically set by USBHS after a USB reset signal finishes and cannot be set by software.<br>This bit is cleared by the following events:<br>– A disconnection condition<br>– Software clears this bit<br>0: Port disabled<br>1: Port enabled |
| 1 | PCD | Port connect detected<br>Set by USBHS when a device connection is detected. This bit can be cleared by writing 1 to this bit. |
| 0 | PCST | Port connect status<br>0: Device is not connected to the port<br>1: Device is connected to the port |

### Host channel-x control register (USBHS_HCHxCTL) (x = 0..15, where x = channel_number)

Address offset: 0x0500 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|



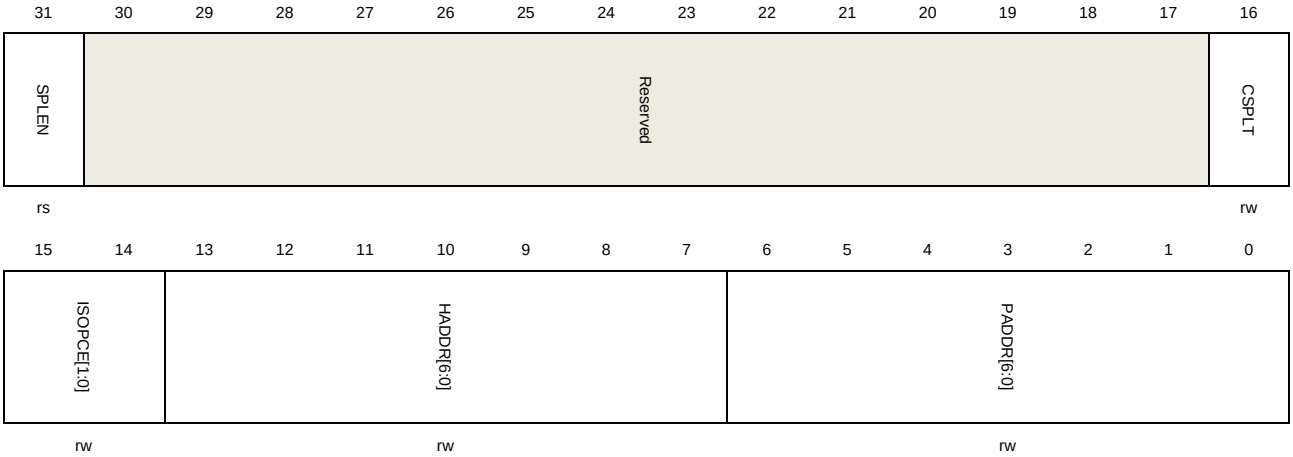

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CEN | Channel enable<br>Set by the application and cleared by USBHS.<br>0: Channel disabled<br>1: Channel enabled<br>Software should follow the operation guide to disable or enable a channel. |
| 30 | CDIS | Channel disable<br>Software can set this bit to disable the channel from processing transactions. Software should follow the operation guide to disable or enable a channel. |
| 29 | ODDFRM | Odd frame<br>For periodic transfers (interrupt or isochronous transfer), this bit controls that whether in an odd frame or even frame this channel's transaction is desired to be processed.<br>0: Even frame<br>1: Odd frame |
| 28:22 | DAR[6:0] | Device address<br>The address of the USB device that this channel wants to communicate with. |
| 21:20 | MPC[1:0] | Multiple Packet Count<br>For periodic transfers, this field indicates to the number of transactions that must be issued per micro-frame. For nonperiodic transfers, it defines how many packets the DMA should fetch or write for this channel before the internal DMA engine changes arbitration.<br>00: Reserved<br>01: 1 transaction to be issued per micro-frame<br>10: 2 transactions to be issued per micro-frame<br>11: 3 transactions to be issued per micro-frame |
| 19:18 | EPTYPE[1:0] | Endpoint type<br>The transfer type of the endpoint that this channel wants to communicate with.<br>00: Control<br>01: Isochronous |

| | | |
|---|---|---|
| | | 10: Bulk<br>11: Interrupt |
| 17 | LSD | Low-Speed device<br>The device that this channel wants to communicate with is a Low-Speed Device. |
| 16 | Reserved | Must be kept at reset value. |
| 15 | EPDIR | Endpoint direction<br>The transfer direction of the endpoint that this channel wants to communicate with.<br>0: OUT<br>1: IN |
| 14:11 | EPNUM[3:0] | Endpoint number<br>The number of the endpoint that this channel wants to communicate with. |
| 10:0 | MPL[10:0] | Maximum packet length<br>The target endpoint's maximum packet length. |

### Host channel-x split transaction control register (USBHS_HCHxSTCTL) (x = 0..15, where x = channel_number)

Address offset: 0x0504 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | SPLEN | Enable High-Speed split transaction<br>Software can set this bit to enable the High-Speed split transaction on this channel. Split transaction is used to initiate a Full-/Low-Speed transaction via the hub and some Full-/Low-Speed device endpoint. |
| 30:17 | Reserved | Must be kept at reset value. |
| 16 | CSPLT | Complete-split enable<br>Software can set this bit to make USBHS perform complete-split transaction, |

| | | |
|---|---|---|
| | | otherwise, USBHS performs start-split transaction. |
| 15:14 | ISOPCE[1:0] | Isochronous OUT Payload Continuation Encoding<br>For Full-Speed isochronous OUT start-split, this field specifies how the High-Speed data payload corresponds to data for a Full-Speed data packet<br>00: High-Speed data is the middle of the Full-Speed data payload<br>01: High-Speed data is the end of the Full-Speed data payload<br>10: High-Speed data is the beginning of the Full-Speed data payload<br>11: High-Speed data is all of the Full-Speed data payload |
| 13:7 | HADDR[6:0] | HUB address<br>This field contains the USB device address of the hub supporting the specified Full-/Low-Speed device for this Full-/Low-Speed transaction. |
| 6:0 | PADDR[6:0] | Port address<br>This field contains the port number of the target hub for this Full-/Low-Speed transaction. |

### Host channel-x interrupt flag register (USBHS_HCHxINTF) (x = 0..15, where x = channel number)

Address offset: 0x0508 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register contains the status and events of a channel, when software gets a channel interrupt, it should read this register for the respective channel to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10 | DTER | Data toggle error<br>The IN transaction gets a data packet but the PID of this packet doesn't match |

| | | |
|---|---|---|
| | | DPID[1:0] bits in USBHS_HCHxLEN register. |
| 9 | REQOVR | Request queue overrun<br>The periodic request queue is full when software starts new transfer. |
| 8 | BBER | Babble error<br>A babble condition occurs on USB bus. A typical reason for babble condition is that a device sends a data packet and the packet length exceeds to the endpoint's maximum packet length. |
| 7 | USBER | USB Bus Error<br>The USB error flag is set when the following conditions occur during receiving a packet:<br>A received packet has a wrong CRC field<br>A stuff error detected on USB bus<br>Timeout when waiting for a response packet |
| 6 | NYET | NYET<br>A NYET response packet received (in High-Speed). |
| 5 | ACK | ACK<br>An ACK response is received or transmitted |
| 4 | NAK | NAK<br>A NAK response is received. |
| 3 | STALL | STALL<br>A STALL response is received. |
| 2 | DMAER | DMA Error<br>An error occurs when DMA tries to fetch or write packet data for this channel. |
| 1 | CH | Channel halted<br>When DMA is not enabled:<br>This channel is disabled by the software request.<br>When DMA is enabled:<br>This channel is disabled by DMA because all the transactions of this channel finish successfully or an USB error occurs. |
| 0 | TF | Transfer finished<br>All the transactions of this channel finish successfully, and no error occurs. For IN channel, this flag will be triggered after PCNT bit in USBHS_HCHxLEN register reaches to zero. For OUT channel, this flag will be triggered when software reads and pops a TF status entry from the RxFIFO. |

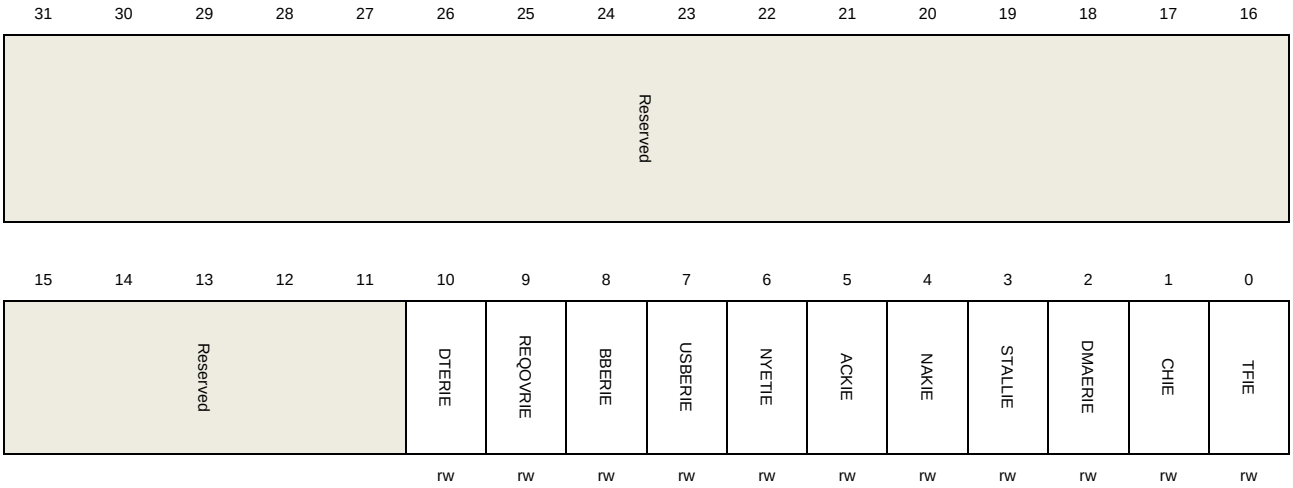
### Host channel-x interrupt enable register (USBHS_HCHxINTEN) (x = 0..15, where x = channel number)

Address offset: 0x050C + (channel_number × 0x20)

Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_HCHxINTF register. If a bit in this register is set by software, the corresponding bit in USBHS_HCHxINTF register is able to trigger a channel interrupt. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)



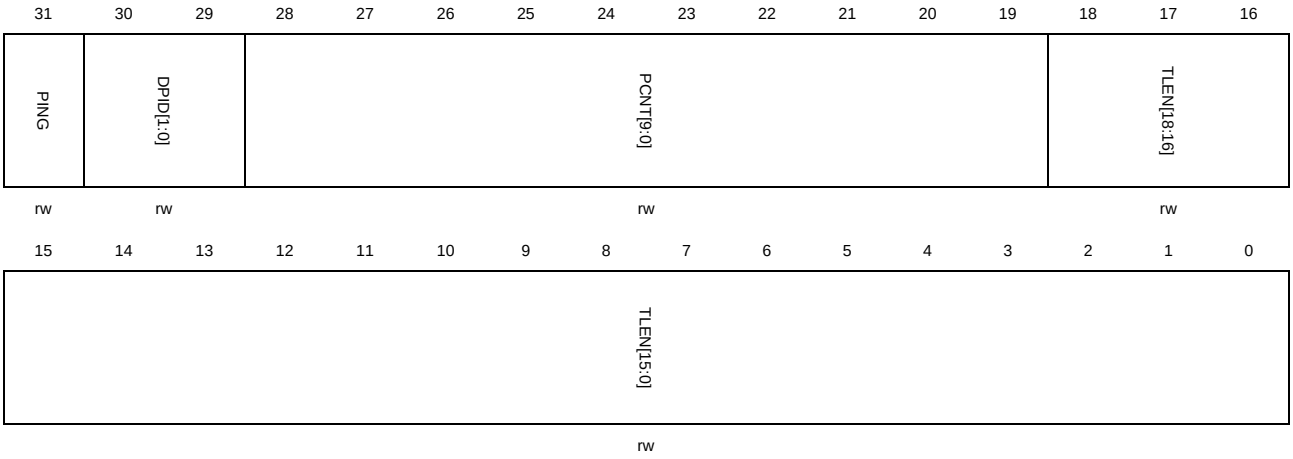

| Bits | Fields | Descriptions |
|---|---|---|
| 31:11 | Reserved | Must be kept at reset value. |
| 10 | DTERIE | Data toggle error interrupt enable<br>0: Disable data toggle error interrupt<br>1: Enable data toggle error interrupt |
| 9 | REQOVRIE | Request queue overrun interrupt enable<br>0: Disable request queue overrun interrupt<br>1: Enable request queue overrun interrupt |
| 8 | BBERIE | Babble error interrupt enable<br>0: Disable babble error interrupt<br>1: Enable babble error interrupt |
| 7 | USBERIE | USB bus error interrupt enable<br>0: Disable USB bus error interrupt<br>1: Enable USB bus error interrupt |
| 6 | NYETIE | NYET interrupt enable<br>0: Disable NYET interrupt<br>1: Enable NYET interrupt |
| 5 | ACKIE | ACK interrupt enable<br>0: Disable ACK interrupt<br>1: Enable ACK interrupt |
| 4 | NAKIE | NAK interrupt enable |

| | | |
|---|---|---|
| | | 0: Disable NAK interrupt<br>1: Enable NAK interrupt |
| 3 | STALLIE | STALL interrupt enable<br>0: Disable STALL interrupt<br>1: Enable STALL interrupt |
| 2 | DMAERIE | DMA Error interrupt enable<br>0: Disable DMA Error interrupt<br>1: Enable DMA Error interrupt |
| 1 | CHIE | Channel halted interrupt enable<br>0: Disable channel halted interrupt<br>1: Enable channel halted interrupt |
| 0 | TFIE | Transfer finished interrupt enable<br>0: Disable transfer finished interrupt<br>1: Enable transfer finished interrupt |

### Host channel-x transfer length register (USBHS_HCHxLEN) (x = 0..15, where x = channel number)

Address offset: 0x0510 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PING | PING token request<br>For OUT transfer, USBHS will perform PING protocol if software sets this bit. USBHS will automatically set this bit when an OUT transaction receives a NAK or NYET handshake. Do not set this bit for IN transfer. |
| 30:29 | DPID[1:0] | Data PID<br>Software should write this field before the transfer starts. For OUT transfer, this field controls the Data PID of the first transmitted packet. For IN transfer, this field controls the |

expected Data PID of the first received packet, and DTERR will be triggered if the Data PID doesn't match. After the transfer starts, USBHS changes and toggles this field automatically following the USB protocol.
00: DATA0
01: DATA2
10: DATA1
11: MDATA (non-control)/SETUP (control)

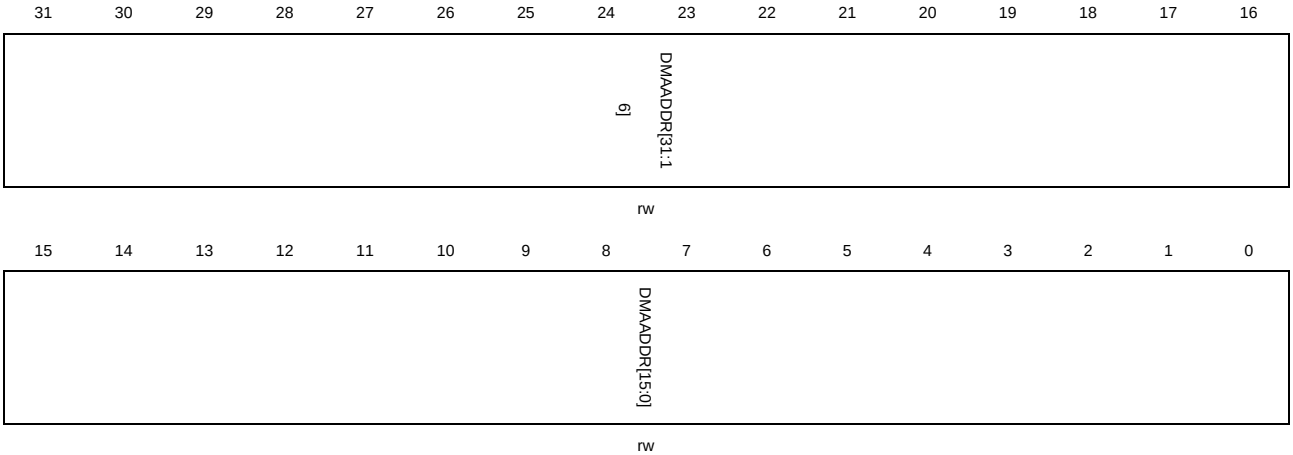
28:19 PCNT[9:0] Packet count
The number of data packets desired to be transmitted (OUT) or received (IN) in transfer. Software should program this field before the channel is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet transmission.

18:0 TLEN[18:0] Transfer length
The total data bytes number of a transfer.
For OUT transfer, this field is the total data bytes of all the data packets desired to be transmitted in an OUT transfer. Software should program this field before the channel is enabled. When software or DMA successfully writes a packet into the channel's data TxFIFO, this field is decreased by the byte size of the packet.
For IN transfer each time software or DMA reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

### Host channel-x DMA address register (USBHS_HCHxDMAADDR) (x = 0..15, where x = channel number)

Address offset: 0x0514 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMAADDR[31:0] | DMA address<br>This field defines the endpoint's DMA address. DMA uses this address to fetch or |

write packet data for this channel.

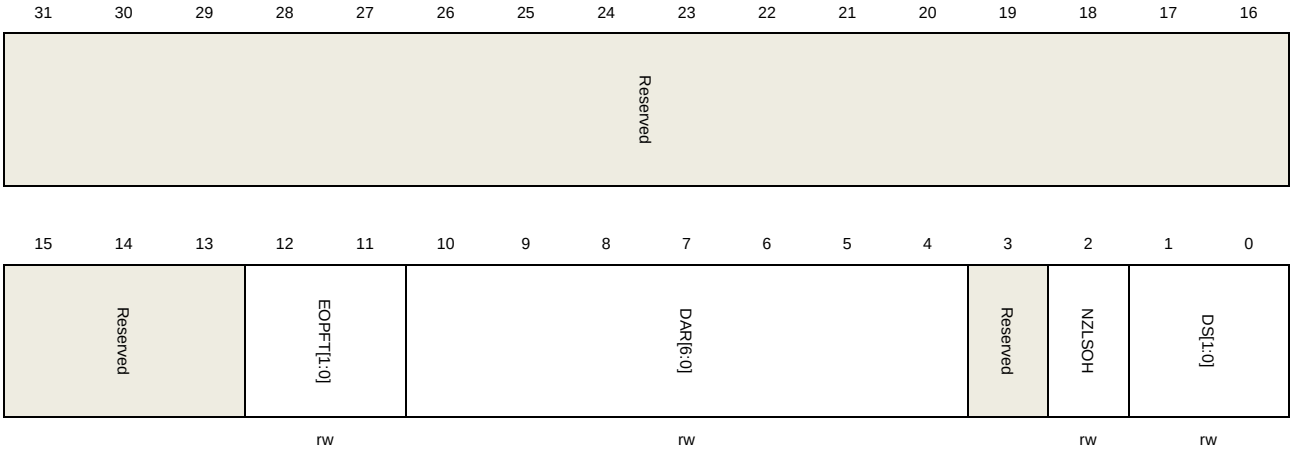
## 49.7.3. Device control and status registers

### Device configuration register (USBHS_DCFG)

Address offset: 0x0800
Reset value: 0x0000 0000

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not change this register after device initialization.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:13 | Reserved | Must be kept at reset value. |
| 12:11 | EOPFT[1:0] | End of periodic frame time<br>This field defines the percentage time point in a frame when the end of periodic frame (EOPF) flag should be triggered.<br>00: 80% of the frame time<br>01: 85% of the frame time<br>10: 90% of the frame time<br>11: 95% of the frame time |
| 10:4 | DAR[6:0] | Device address<br>This field defines the USB device's address. USBHS uses this field to match with the incoming token's device address field. Software should program this field after receiving a Set Address command from USB host. |
| 3 | Reserved | Must be kept at reset value. |
| 2 | NZLSOH | Non-zero-length status OUT handshake<br>When a USB device receives a non-zero-length data packet during status OUT stage, this field controls that USBHS should receive this packet or reject this packet with a STALL handshake. |

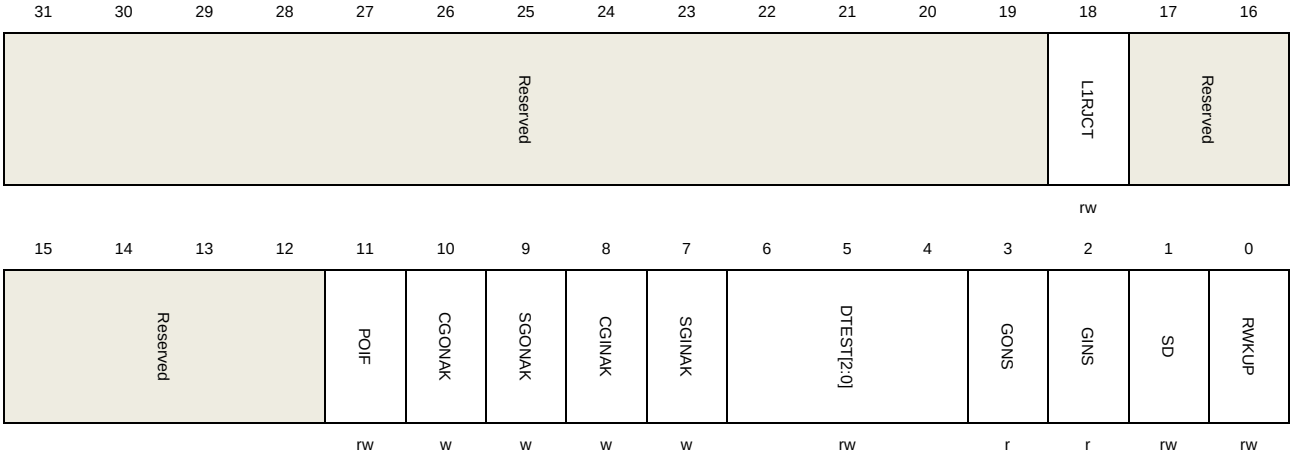
| | | |
|---|---|---|
| | | 0: Treat this packet as a normal packet and response according to the status of NAKS and STALL bits in USBHS_DOEPxCTL register.<br>1: Send a STALL handshake and don't save the received OUT packet. |
| 1:0 | DS[1:0] | Device speed<br>This field controls the device speed when the device is connected to a host.<br>00: High-Speed<br>01: Full-Speed<br>Others: Reserved |

### Device control register (USBHS_DCTL)

Address offset: 0x0804

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:19 | Reserved | Must be kept at reset value. |
| 18 | L1RJCT | Deep sleep reject<br>When this bit is set, the core response NYET for LPM transaction with BESL greater than BSELTH. |
| 17:12 | Reserved | Must be kept at reset value. |
| 11 | POIF | Power-on initialization finished<br>Software should set this bit to notify USBHS that the registers are initialized after waking up from power off state. |
| 10 | CGONAK | Clear global OUT NAK<br>Software sets this bit to clear GONS bit in this register. |
| 9 | SGONAK | Set global OUT NAK<br>Software sets this bit to set GONS bit in this register.<br>When GONS bit is zero, setting this bit will also cause GONAK flag in |

| | | |
|---|---|---|
| | | USBHS_GINTF register triggered after a while. Software should clear the GONAK flag before writing this bit again. |
| 8 | CGINAK | Clear global IN NAK<br>Software sets this bit to clear GINS bit in this register. |
| 7 | SGINAK | Set global IN NAK<br>Software sets this bit to set GINS bit in this register.<br>When GINS bit is zero, setting this bit will also cause GINAK flag in USBHS_GINTF register triggered after a while. Software should clear the GINAK flag before writing this bit again. |
| 6:4 | DTEST[2:0] | Device Test control<br>The software writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is sended on the port. When test mode is used, the HS_CUR_FE bit in USBHS_GUSBCS register should also be set.<br>000: Test mode disabled<br>001: Test_J mode<br>010: Test_K mode<br>011: Test_SE0_NAK mode<br>100: Test_Packet mode<br>101: Test_Force_Enable<br>Others: Reserved |
| 3 | GONS | Global OUT NAK status<br>0: The handshake that USBHS response to OUT transaction packet and whether to save the OUT data packet are decided by Rx FIFO status, endpoint's NAK and STALL bits.<br>1: USHBS always responses to OUT transaction with NAK handshake and doesn't save the incoming OUT data packet. |
| 2 | GINS | Global IN NAK status<br>0: The response to IN transaction is decided by Tx FIFO status, endpoint's NAK and STALL bits.<br>1: USHBS always responses to IN transaction with a NAK handshake. |
| 1 | SD | Soft disconnect<br>Software can use this bit to generate a soft disconnect condition on USB bus. After this bit is set, USBHS first falls back to Full-Speed if currently operating at High-Speed, and then switches off the pull up resistor on DP line. This will cause the host to detect a device disconnect.<br>0: No soft disconnect generated.<br>1: Generate a soft disconnect. |
| 0 | RWKUP | Remote wakeup<br>In suspend state, software can use this bit to generate a Remote wake up signal to inform host that it should resume the USB bus.<br>0: No remote wakeup signal generated. |

1: Generate remote wakeup signal.

When the core is LPM enabled and in sleep status, if this bit is set, the core continues to drive it and clears it automatically after 50us (TL1DevDrvResume). The application cannot set this bit when bRemoteWake value received from LPM transaction is zero.

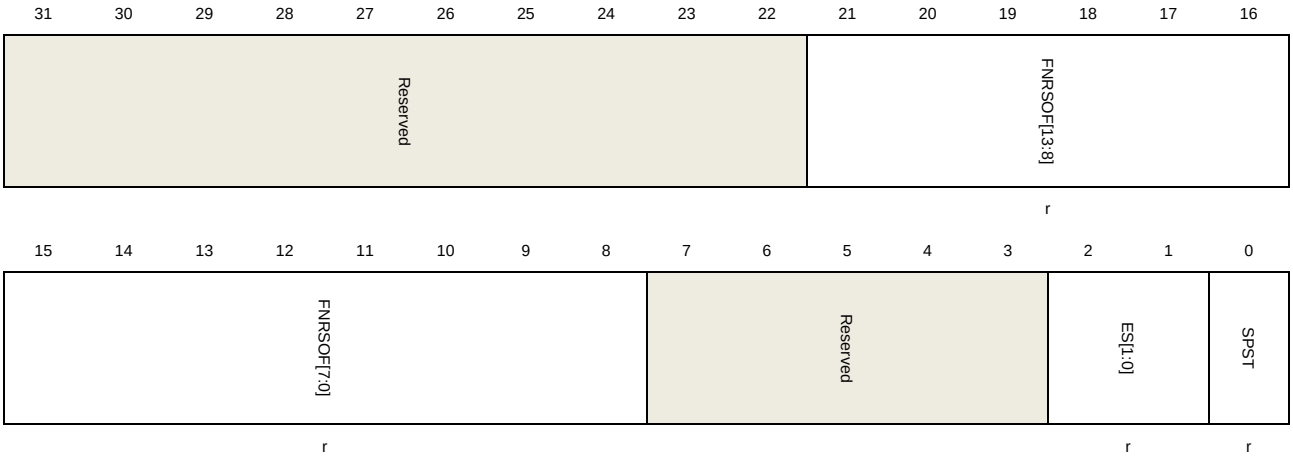
### Device status register (USBHS_DSTAT)

Address offset: 0x0808

Reset value: 0x0000 0000

This register contains status and information of the USBHS in device mode.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:22 | Reserved | Must be kept at reset value. |
| 21:8 | FNRSOF[13:0] | The frame number of the received SOF.<br>USBHS always update this field after receiving a SOF token. |
| 7:3 | Reserved | Must be kept at reset value. |
| 2:1 | ES[1:0] | Enumerated speed<br>This field reports the enumerated device speed. Software should read this field after the ENUMF flag in USBHS_GINTF register is triggered.<br>00: High-Speed<br>01: Full-Speed<br>Others: reserved |
| 0 | SPST | Suspend status<br>This bit reports whether device is in suspend state.<br>0: Device is not in suspend state.<br>1: Device is in suspend state. |

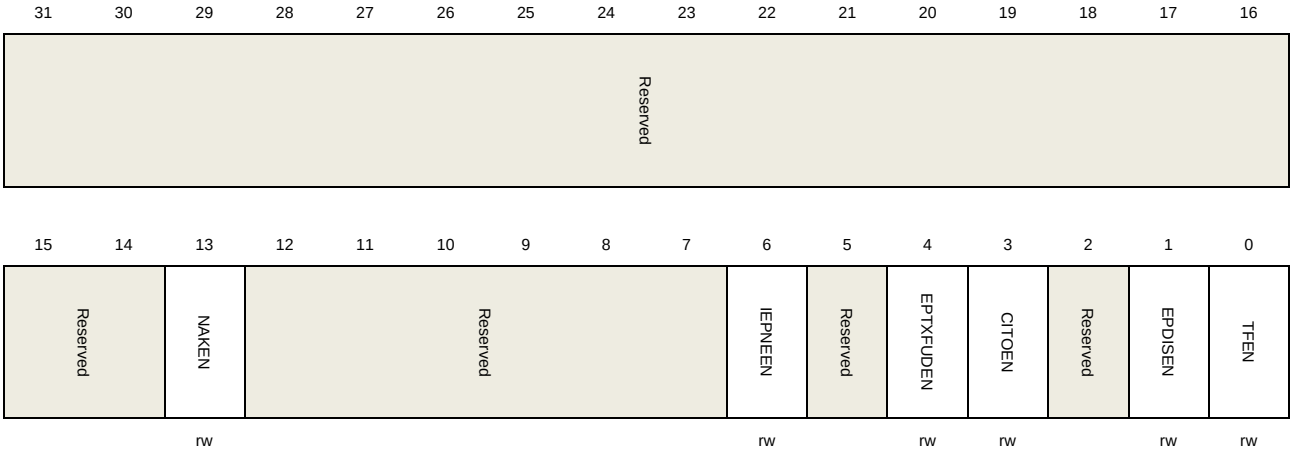
### Device IN endpoint common interrupt enable register (USBHS_DIEPINTEN)

Address offset: 0x810

Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_DIEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DIEPxINTF register is able to trigger an endpoint interrupt in USBHS_DAEPINT register. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | NAKEN | NAK handshake sent by USBHS interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 12:7 | Reserved | Must be kept at reset value. |
| 6 | IEPNEEN | IN endpoint NAK effective interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 5 | Reserved | Must be kept at reset value. |
| 4 | EPTXFUDEN | Endpoint Tx FIFO underrun interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 3 | CITOEN | Control In Timeout interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 2 | Reserved | Must be kept at reset value. |
| 1 | EPDISEN | Endpoint disabled interrupt enable bit |

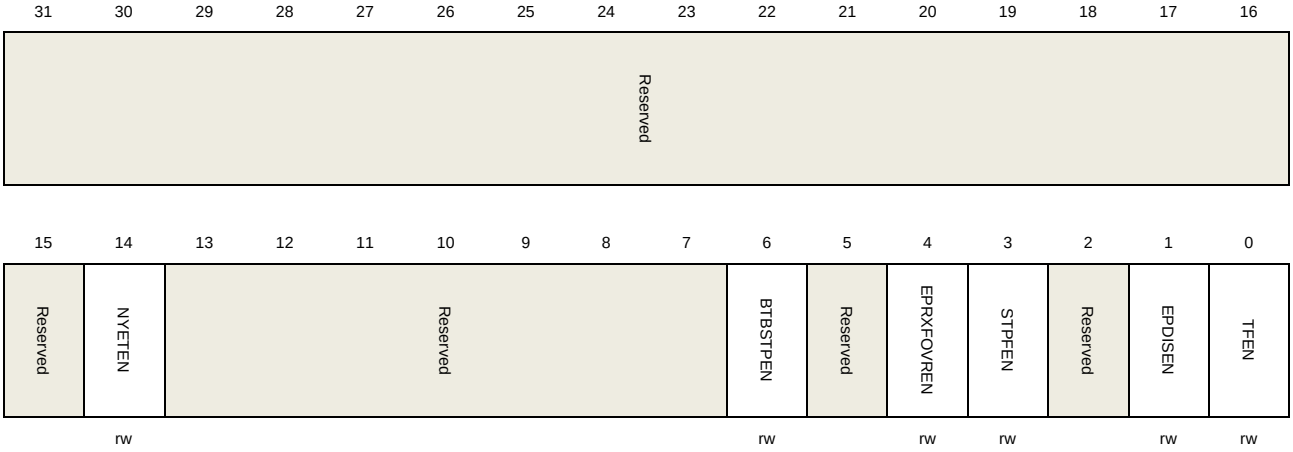
| | | |
|---|---|---|
| | | 0: Disable interrupt<br>1: Enable interrupt |
| 0 | TFEN | Transfer finished interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |

### Device OUT endpoint common interrupt enable register (USBHS_DOEPINTEN)

Address offset: 0x0814
Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_DOEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DOEPxINTF register is able to trigger an endpoint interrupt in USBHS_DAEPINT register. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | NYETEN | Send NYET handshake interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 13:7 | Reserved | Must be kept at reset value. |
| 6 | BTBSTPEN | Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 5 | Reserved | Must be kept at reset value. |
| 4 | EPRXFOVREN | Endpoint Rx FIFO overrun interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |

| | | |
|---|---|---|
| 3 | STPFEN | SETUP phase finished (Only for control OUT endpoint) interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 2 | Reserved | Must be kept at reset value. |
| 1 | EPDISEN | Endpoint disabled interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 0 | TFEN | Transfer finished interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |

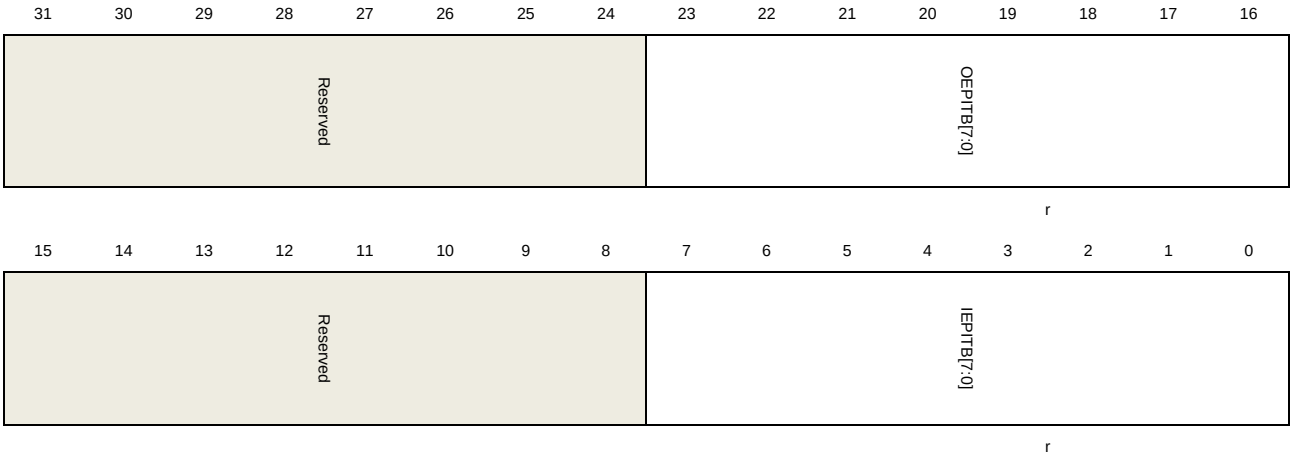
### Device all endpoints interrupt register (USBHS_DAEPINT)

Address offset: 0x0818

Reset value: 0x0000 0000

When an endpoint interrupt is triggered, USBHS sets corresponding bit in this register and software should read this register to know which endpoint is asserting an interrupt.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | OEPITB[7:0] | Device all OUT endpoints interrupt bits<br>Each bit represents an OUT endpoint:<br>Bit 16 for OUT endpoint 0, bit 23 for OUT endpoint 7. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | IEPITB[7:0] | Device all IN endpoints interrupt bits<br>Each bit represents an IN endpoint:<br>Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7. |

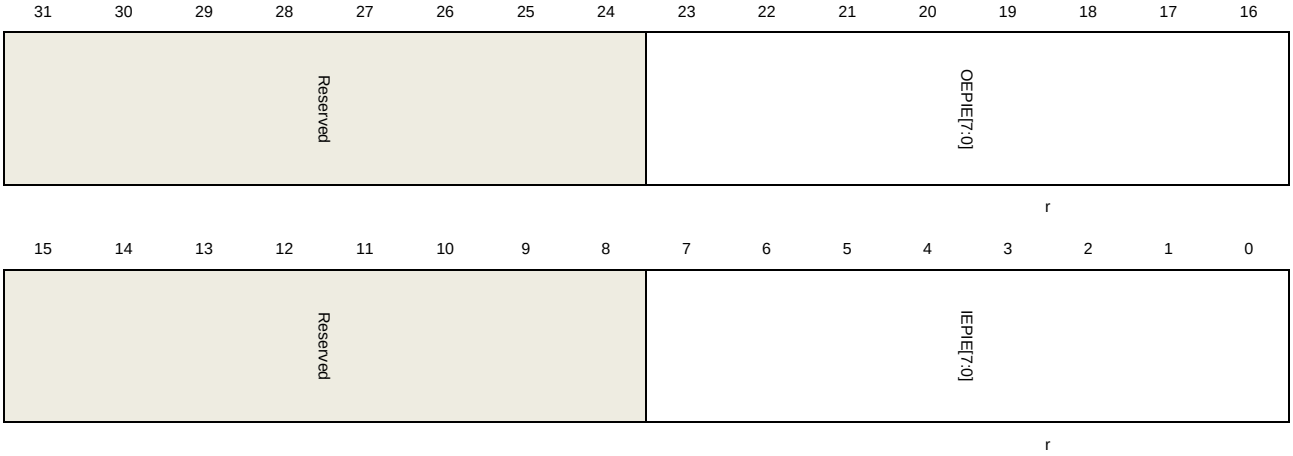
### Device all endpoints interrupt enable register (USBHS_DAEPINTEN)

Address offset: 0x081C

Reset value: 0x0000 0000

This register can be used by software to enable or disable an endpoint's interrupt. Only the endpoint whose corresponding bit in this register is set is able to cause the endpoint interrupt flag OEPIF or IEPIF in USBHS_GINTF register.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23:16 | OEPIE[7:0] | Out endpoint interrupt enable<br>0: Disable OUT endpoint-n interrupt<br>1: Enable OUT endpoint-n interrupt<br>Each bit represents an OUT endpoint:<br>Bit 16 for OUT endpoint 0, bit 23 for OUT endpoint 7. |
| 15:8 | Reserved | Must be kept at reset value. |
| 7:0 | IEPIE[7:0] | IN endpoint interrupt enable bits<br>0: Disable IN endpoint-n interrupt<br>1: Enable IN endpoint-n interrupt<br>Each bit represents an IN endpoint:<br>Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7. |

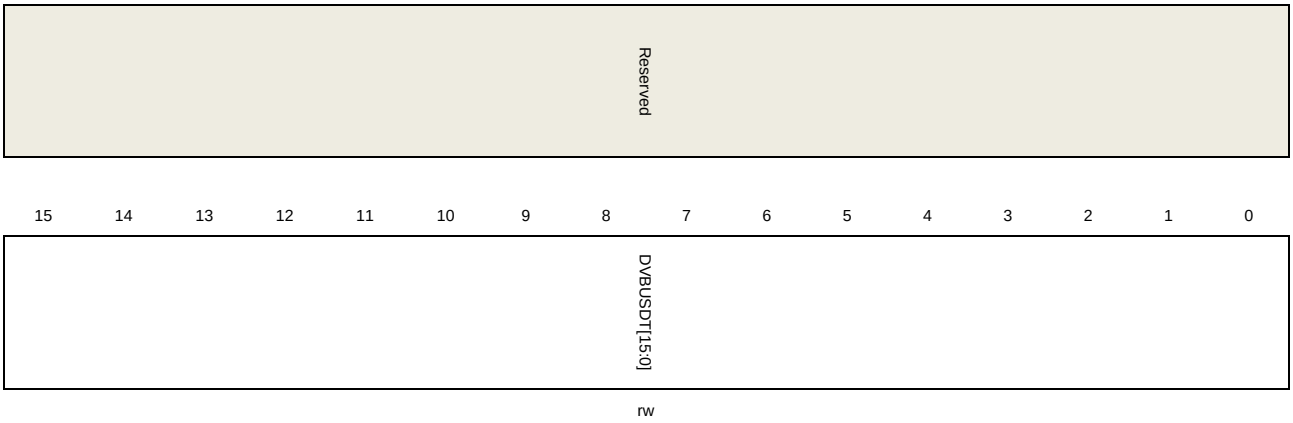
### Device VBUS discharge time register (USBHS_DVBUSDT)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register has to be accessed by word (32-bit)





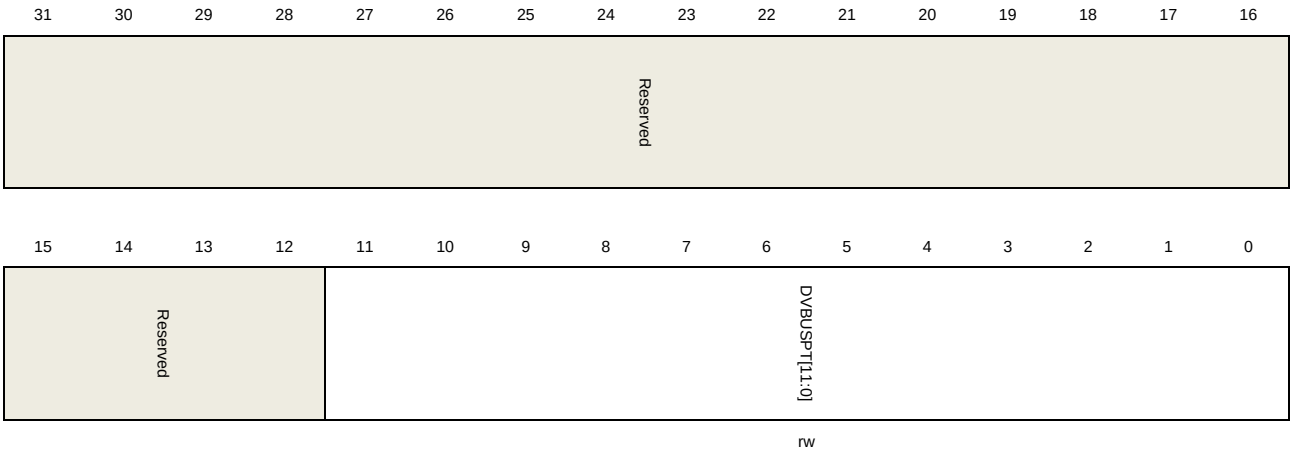

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | DVBUSDT[15:0] | Device $V_{BUS}$ discharge time<br>There is a discharge process after $V_{BUS}$ pulsing in SRP protocol. This field defines the discharge time of $V_{BUS}$. The true discharge time is 1024*DVBUSDT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock. |

### Device VBUS pulsing time register (USBHS_DVBUSPT)

Address offset: 0x082C

Reset value: 0x0000 05B8

This register has to be accessed by word (32-bit)



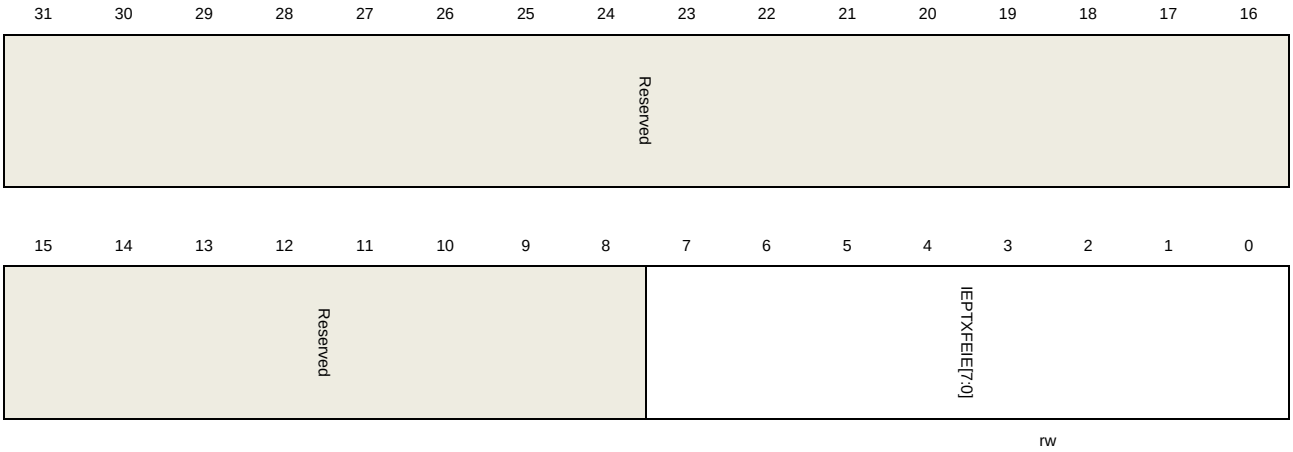

| Bits | Fields | Descriptions |
|---|---|---|
| 31:12 | Reserved | Must be kept at reset value. |
| 11:0 | DVBUSPT[11:0] | Device $V_{BUS}$ pulsing time<br>This field defines the pulsing time for $V_{BUS}$. The true pulsing time is 1024*DVBUSPT[11:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock. |

### Device IN endpoint FIFO empty interrupt enable register (USBHS_DIEPFEINTEN)

Address offset: 0x0834
Reset value: 0x0000 0000

This register contains the enabled bits for the Tx FIFO empty interrupts of IN endpoints.

This register has to be accessed by word (32-bit)



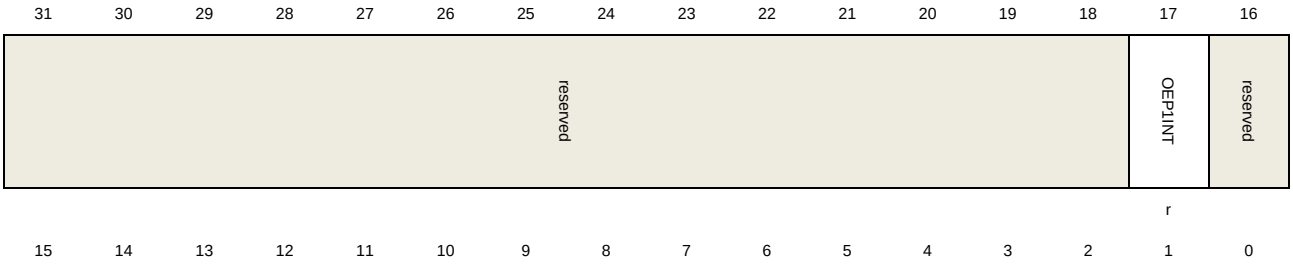

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7:0 | IEPTXFEIE[7:0] | IN endpoint Tx FIFO empty interrupt enable bits<br>This field controls whether the TXFE bit in USBHS_DIEPxINTF register is able to generate an endpoint interrupt bit in USBHS_DAEPINT register.<br>Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7<br>0: Disable FIFO empty interrupt<br>1: Enable FIFO empty interrupt |

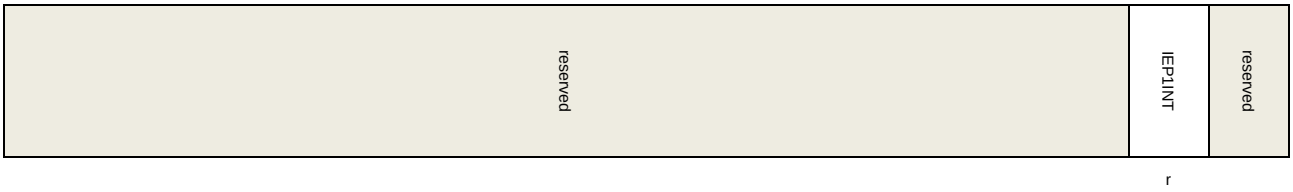
### Device endpoint 1 interrupt register (USBHS_DEP1INT)

Address offset: 0x0838
Reset value: 0x0000 0000

When ep1 out or in interrupt is triggered, USBHS sets corresponding bit in this register and software should read this register to know which endpoint is asserting the ep1 interrupt.

This register has to be accessed by word (32-bit)





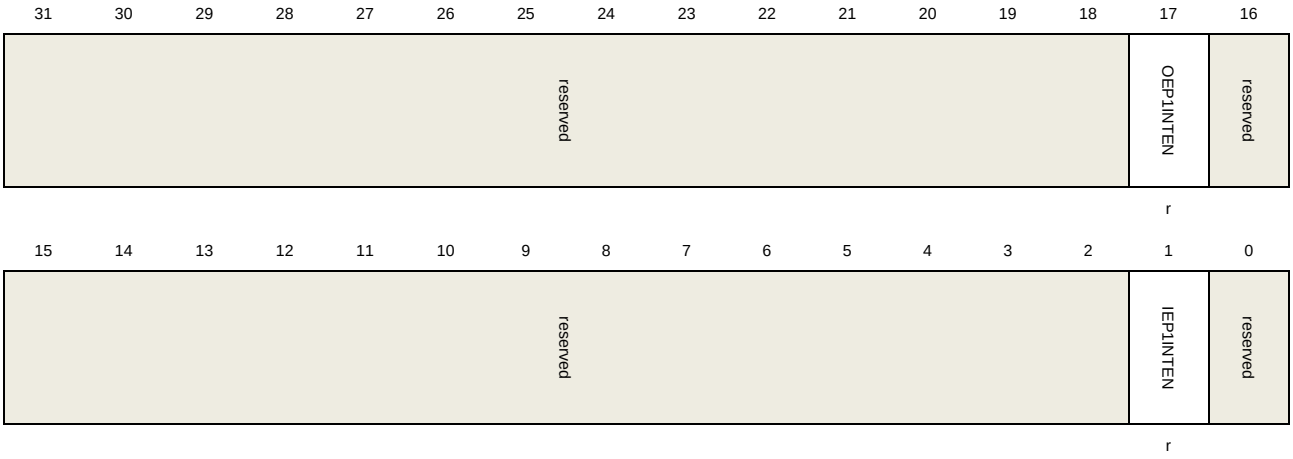

| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | OEP1INT | OUT Endpoint 1 interrupt |
| 16:2 | Reserved | Must be kept at reset value. |
| 1 | IEP1INT | IN Endpoint 1 interrupt |
| 0 | Reserved | Must be kept at reset value. |

### Device endpoint 1 interrupt enable register (USBHS_DEP1INTEN)

Address offset: 0x083C
Reset value: 0x0000 0000

This register can be used by software to enable or disable endpoint-1's interrupt. Only the endpoint whose corresponding bit in this register is set is able to cause the endpoint-1 in or out interrupt.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:18 | Reserved | Must be kept at reset value. |
| 17 | OEP1INTEN | OUT Endpoint 1 interrupt enable |
| 16:2 | Reserved | Must be kept at reset value. |
| 1 | IEP1INTEN | IN Endpoint 1 interrupt enable |
| 0 | Reserved | Must be kept at reset value. |

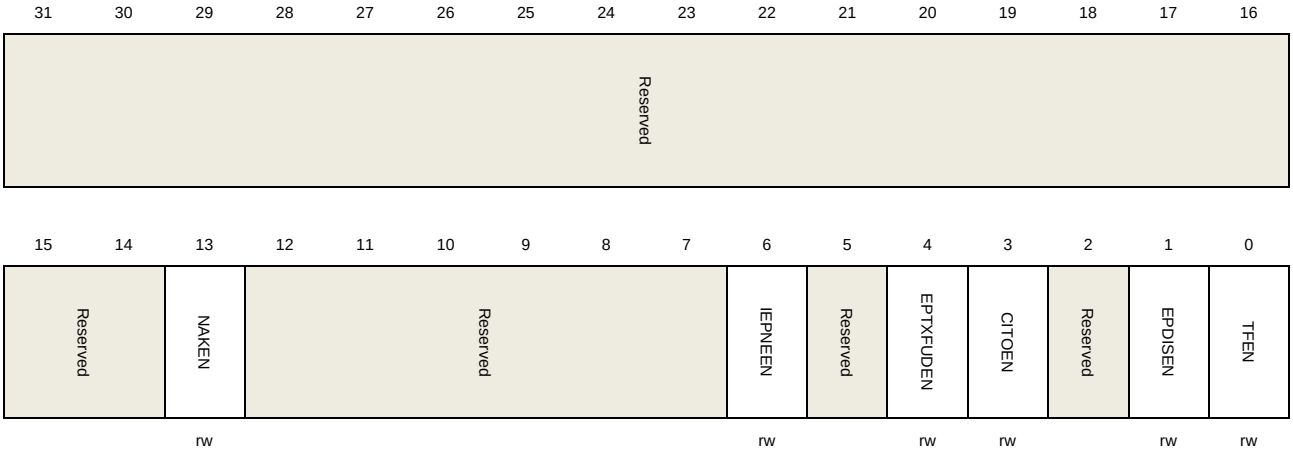
### Device IN endpoint-1 interrupt enable register (USBHS_DIEP1INTEN)

Address offset: 0x844

Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBHS_DIEP1INTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DIEP1INTF register is able to trigger an endpoint interrupt in USBHS_DEP1INT register. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | NAKEN | Interrupt enable bit of NAK handshake sent by USBHS<br>0: Disable interrupt<br>1: Enable interrupt |
| 12:7 | Reserved | Must be kept at reset value. |
| 6 | IEPNEEN | IN endpoint NAK effective interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 5 | Reserved | Must be kept at reset value. |
| 4 | EPTXFUDEN | Endpoint Tx FIFO underrun interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 3 | CITOEN | Control In Timeout interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 2 | Reserved | Must be kept at reset value. |
| 1 | EPDISEN | Endpoint disabled interrupt enable bit |

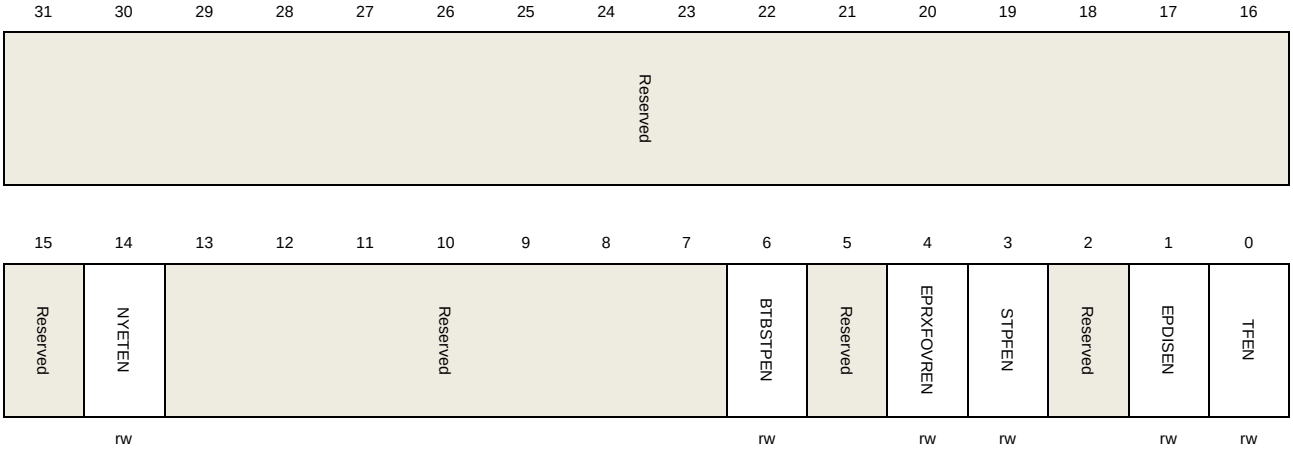
| | | |
|---|---|---|
| | | 0: Disable interrupt<br>1: Enable interrupt |
| 0 | TFEN | Transfer finished interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |

### Device OUT endpoint-1 interrupt enable register (USBHS_DOEP1INTEN)

Address offset: 0x0884
Reset value: 0x0000 0000

This register contains the interrupt enabled bits for the flags in USBHS_DOEP1INTF register. If a bit in this register is set by software, the corresponding bit in USBHS_DOEP1INTF register is able to trigger an endpoint interrupt in USBHS_DEP1INT register. The bits in this register are set and cleared by software.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | NYETEN | Send NYET handshake interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 13:7 | Reserved | Must be kept at reset value. |
| 6 | BTBSTPEN | Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 5 | Reserved | Must be kept at reset value. |
| 4 | EPRXFOVREN | Endpoint Rx FIFO overrun interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |

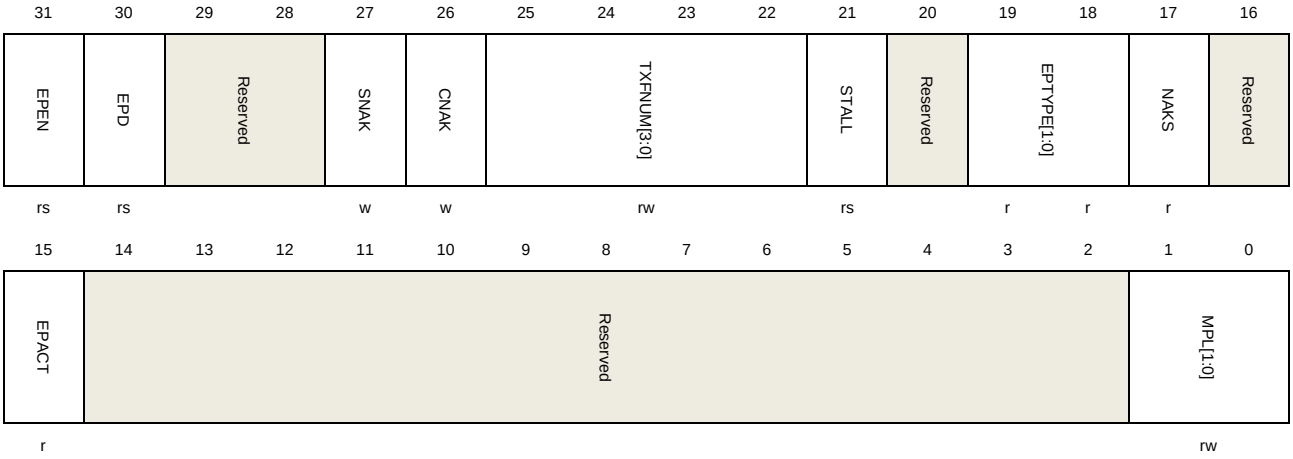
| | | |
|---|---|---|
| 3 | STPFEN | SETUP phase finished (Only for control OUT endpoint) interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 2 | Reserved | Must be kept at reset value. |
| 1 | EPDISEN | Endpoint disabled interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |
| 0 | TFEN | Transfer finished interrupt enable bit<br>0: Disable interrupt<br>1: Enable interrupt |

### Device IN endpoint 0 control register (USBHS_DIEP0CTL)

Address offset: 0x0900

Reset value: 0x0000 8000

This register has to be accessed by word (32-bit)



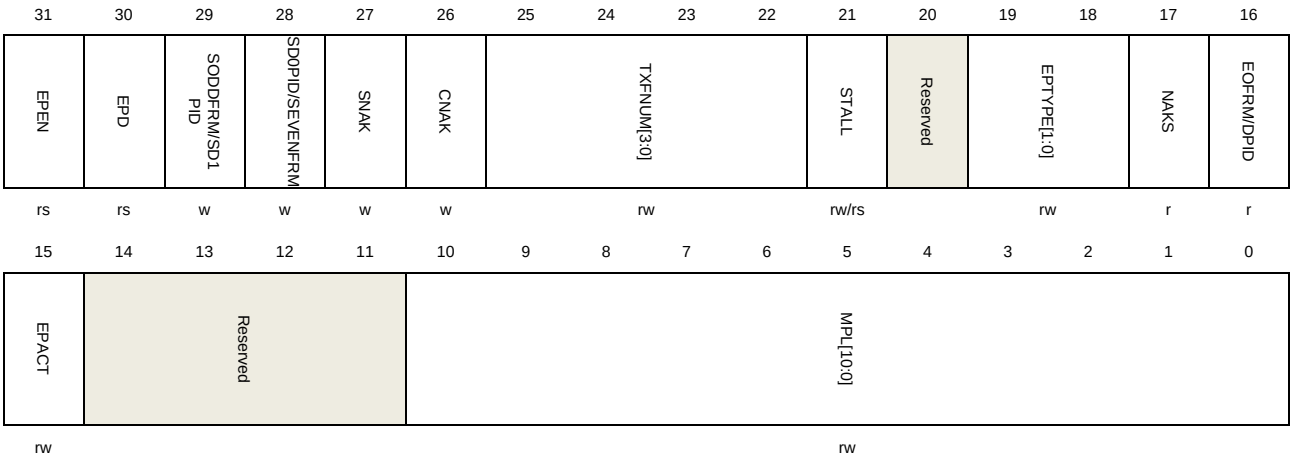

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | EPEN | Endpoint enable<br>Set by the application and cleared by USBHS.<br>0: Endpoint disabled<br>1: Endpoint enabled<br>Software should follow the operation guide to disable or enable an endpoint. |
| 30 | EPD | Endpoint disable<br>Software can set this bit to disable the endpoint. Software should follow the operation guide to disable or enable an endpoint. |
| 29:28 | Reserved | Must be kept at reset value. |
| 27 | SNAK | Set NAK<br>Software sets this bit to set NAKS bit in this register. |

| | | |
|---|---|---|
| 26 | CNAK | Clear NAK<br>Software sets this bit to clear NAKS bit in this register. |
| 25:22 | TXFNUM[3:0] | Tx FIFO number<br>Define the Tx FIFO number of IN endpoint 0. |
| 21 | STALL | STALL handshake<br>Software can set this bit to make USBHS send STALL handshake when receiving IN token. USBHS will clear this bit after a SETUP token is received on the corresponding OUT endpoint 0. This bit has a higher priority than NAKS bit in this register and GINS bit in USBHS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect. |
| 20 | Reserved | Must be kept at reset value. |
| 19:18 | EPTYPE[1:0] | Endpoint type<br>This field is fixed to '00' for control endpoint. |
| 17 | NAKS | NAK status<br>This bit controls the NAK status of USBHS when both STALL bit in this register and GINS bit in USBHS_DCTL register are cleared:<br>0: USBHS sends data or handshake packets according to the status of the endpoint's Tx FIFO.<br>1: USBHS always sends NAK handshake to the IN token.<br>This bit is read-only and software should use CNAK and SNAK in this register to control this bit. |
| 16 | Reserved | Must be kept at reset value. |
| 15 | EPACT | Endpoint active<br>This field is fixed to '1' for endpoint 0. |
| 14:2 | Reserved | Must be kept at reset value. |
| 1:0 | MPL[1:0] | Maximum packet length<br>This field defines the maximum packet length for a control data packet. As described in USB 2.0 protocol, there are 4 kinds of length for control transfers:<br>00: 64 bytes<br>01: 32 bytes<br>10: 16 bytes<br>11: 8 bytes |

### Device IN endpoint-x control register (USBHS_DIEPxCTL) (x = 1..7, where x = endpoint_number)

Address offset: 0x0900 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | EPEN | Endpoint enable<br>Set by the application and cleared by USBHS.<br>0: Endpoint disabled<br>1: Endpoint enabled<br>Software should follow the operation guide to disable or enable an endpoint. |
| 30 | EPD | Endpoint disable<br>Software can set this bit to disable the endpoint. Software should following the operation guide to disable or enable an endpoint. |
| 29 | SODDFRM | Set odd frame (For isochronous IN endpoints)<br>This bit has effect only if this is an isochronous IN endpoint.<br>Software sets this bit to set EOFRM bit in this register. |
| | SD1PID | Set DATA1 PID (For interrupt/bulk IN endpoints)<br>Software sets this bit to set DPID bit in this register. |
| 28 | SEVENFRM | Set even frame (For isochronous IN endpoints)<br>Software sets this bit to clear EOFRM bit in this register. |
| | SD0PID | Set DATA0 PID (For interrupt/bulk IN endpoints)<br>Software sets this bit to clear DPID bit in this register. |
| 27 | SNAK | Set NAK<br>Software sets this bit to set NAKS bit in this register. |
| 26 | CNAK | Clear NAK<br>Software sets this bit to clear NAKS bit in this register. |
| 25:22 | TXFNUM[3:0] | Tx FIFO number<br>Defines the Tx FIFO number of this IN endpoint. |
| 21 | STALL | STALL handshake<br>Software can set this bit to make USBHS send STALL handshake when receiving IN token. This bit has a higher priority than NAKS bit in this register and GINS bit in USBHS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes |

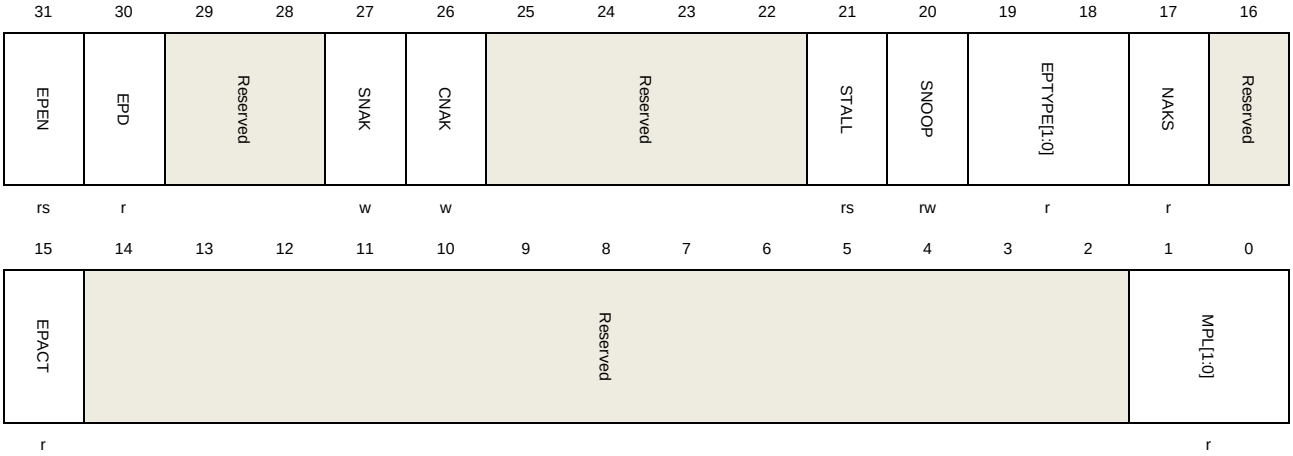
| | | |
|---|---|---|
| | | effect.<br>For control IN endpoint:<br>Only USBHS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it.<br>For interrupt or bulk IN endpoint:<br>Only software can clear this bit |
| 20 | Reserved | Must be kept at reset value. |
| 19:18 | EPTYPE[1:0] | Endpoint type<br>This field defines the transfer type of this endpoint:<br>00: Control<br>01: Isochronous<br>10: Bulk<br>11: Interrupt |
| 17 | NAKS | NAK status<br>This bit controls the NAK status of USBHS when both STALL bit in this register and GINS bit in USBHS_DCTL register are cleared:<br>0: USBHS sends data or handshake packets according to the status of the endpoint's Tx FIFO.<br>1: USBHS always sends NAK handshake to the IN token.<br>This bit is read-only and software should use CNAK and SNAK in this register to control this bit. |
| 16 | EOFRM | Even/odd frame (For isochronous IN endpoints)<br>For isochronous transfer, software can use this bit to control that USBHS only sends data packets for IN tokens in even or odd frames. If the current frame number's parity doesn't match with this bit, USBHS only responses with a zero-length packet.<br>0: Only sends data in even frames<br>1: Only sends data in odd frames |
| | DPID | Endpoint data PID (For interrupt/bulk IN endpoints)<br>These is a data PID toggle scheme in interrupt or bulk transfer. Software should set SD0PID to set this bit before a transfer starts and USBHS maintains this bit during transfers by following the data toggle scheme described in USB protocol.<br>0: Data packet's PID is DATA0<br>1: Data packet's PID is DATA1 |
| 15 | EPACT | Endpoint active<br>This bit controls whether this endpoint is active. If an endpoint is not active, it ignores all tokens and doesn't make any response. |
| 14:11 | Reserved | Must be kept at reset value. |
| 10:0 | MPL[10:0] | This field defines the maximum packet length in bytes. |

### Device OUT endpoint 0 control register (USBHS_DOEP0CTL)

Address offset: 0x0B00

Reset value: 0x0000 8000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | EPEN | Endpoint enable<br>Set by the application and cleared by USBHS.<br>0: Endpoint disabled<br>1: Endpoint enabled<br>Software should follow the operation guide to disable or enable an endpoint. |
| 30 | EPD | Endpoint disable<br>This bit is fixed to 0 for OUT endpoint 0. |
| 29:28 | Reserved | Must be kept at reset value. |
| 27 | SNAK | Set NAK<br>Software sets this bit to set NAKS bit in this register. |
| 26 | CNAK | Clear NAK<br>Software sets this bit to clear NAKS bit in this register |
| 25:22 | Reserved | Must be kept at reset value. |
| 21 | STALL | STALL handshake<br>Software can set this bit to make USBHS send STALL handshake during an OUT transaction. USBHS will clear this bit after a SETUP token is received on OUT endpoint 0. This bit has a higher priority than NAKS bit in this register, i.e. if both STALL and NAKS bits are set, the STALL bit takes effect. |
| 20 | SNOOP | Snoop mode<br>This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBHS doesn't check the received data packet's CRC value.<br>0: Snoop mode disabled |

| | | |
|---|---|---|
| | | 1: Snoop mode enabled |
| 19:18 | EPTYPE[1:0] | Endpoint type<br>This field is fixed to '00' for control endpoint. |
| 17 | NAKS | NAK status<br>This bit controls the NAK status of USBHS when both STALL bit in this register and GONS bit in USBHS_DCTL register are cleared:<br>0: USBHS sends data or handshake packets according to the status of the endpoint's Rx FIFO.<br>1: USBHS always sends NAK handshake to the OUT token.<br>This bit is read-only and software should use CNAK and SNAK in this register to control this bit. |
| 16 | Reserved | Must be kept at reset value. |
| 15 | EPACT | Endpoint active<br>This field is fixed to '1' for endpoint 0. |
| 14:2 | Reserved | Must be kept at reset value. |
| 1:0 | MPL[1:0] | Maximum packet length<br>This is a read-only field, and its value comes from the MPL field of USBHS_DIEP0CTL register:<br>00: 64 bytes<br>01: 32 bytes<br>10: 16 bytes<br>11: 8 bytes |

### Device OUT endpoint-x control register (USBHS_DOEPxCTL) (x = 1..7, where x = endpoint_number)

Address offset: 0x0B00 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

The application uses this register to control the operation of each logical OUT endpoint other than OUT endpoint 0.

This register has to be accessed by word (32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EPEN | EPD | SODDFRM/SD1PID | SEVENFRM/SD0PID | SNAK | CNAK | Reserved | | | | STALL | SNOOP | EPTYPE[1:0] | | NAKS | EOFRM/DPID |
| rs | rs | w | w | w | w | | | | | rw/rs | rw | rw | | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| EPACT | Reserved | MPL[10:0] |
|---|---|---|
| rw | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | EPEN | Endpoint enable<br>Set by the application and cleared by USBHS.<br>0: Endpoint disabled<br>1: Endpoint enabled<br>Software should follow the operation guide to disable or enable an endpoint. |
| 30 | EPD | Endpoint disable<br>Software can set this bit to disable the endpoint. Software should follow the operation guide to disable or enable an endpoint. |
| 29 | SODDFRM | Set odd frame (For isochronous OUT endpoints)<br>This bit has effect only if this is an isochronous OUT endpoint.<br>Software sets this bit to set EOFRM bit in this register. |
| | SD1PID | Set DATA1 PID (For interrupt/bulk OUT endpoints)<br>Software sets this bit to set DPID bit in this register. |
| 28 | SEVENFRM | Set even frame (For isochronous OUT endpoints)<br>Software sets this bit to clear EOFRM bit in this register. |
| | SD0PID | Set DATA0 PID (For interrupt/bulk OUT endpoints)<br>Software sets this bit to clear DPID bit in this register. |
| 27 | SNAK | Set NAK<br>Software sets this bit to set NAKS bit in this register. |
| 26 | CNAK | Clear NAK<br>Software sets this bit to clear NAKS bit in this register. |
| 25:22 | Reserved | Must be kept at reset value. |
| 21 | STALL | STALL handshake<br>Software can set this bit to make USBHS send STALL handshake during an OUT transaction. This bit has a higher priority than NAKS bit in this register and GINS in USBHS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect.<br>For control OUT endpoint:<br>Only USBHS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it.<br>For interrupt or bulk OUT endpoint:<br>Only software can clear this bit. |
| 20 | SNOOP | Snoop mode |

| | | |
|---|---|---|
| | | This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBHS doesn't check the received data packet's CRC value.<br>0: Snoop mode disabled<br>1: Snoop mode enabled |
| 19:18 | EPTYPE[1:0] | Endpoint type<br>This field defines the transfer type of this endpoint:<br>00: Control<br>01: Isochronous<br>10: Bulk<br>11: Interrupt |
| 17 | NAKS | NAK status<br>This bit controls the NAK status of USBHS when both STALL bit in this register and GONS bit in USBHS_DCTL register are cleared:<br>0: USBHS sends handshake packets according to the status of the endpoint's Rx FIFO.<br>1: USBHS always sends NAK handshake to the OUT token.<br>This bit is read-only and software should use CNAK and SNAK in this register to control this bit. |
| 16 | EOFRM | Even/odd frame (For isochronous OUT endpoints)<br>For isochronous transfer, software can use this bit to control that USBHS only receives data packets in even or odd frames. If the current frame number's parity doesn't match with this bit, USBHS just drops the data packet.<br>0: Only sends data in even frames<br>1: Only sends data in odd frames |
| | DPID | Endpoint data PID (For interrupt/bulk OUT endpoints)<br>These is a data PID toggle scheme in interrupt or bulk transfer. Software should set SD0PID to set this bit before a transfer starts and USBHS maintains this bit during transfer following the data toggle scheme described in USB protocol.<br>0: Data packet's PID is DATA0<br>1: Data packet's PID is DATA1 |
| 15 | EPACT | Endpoint active<br>This bit controls whether this endpoint is active. If an endpoint is not active, it ignores all tokens and doesn't make any response. |
| 14:11 | Reserved | Must be kept at reset value. |
| 10:0 | MPL[10:0] | This field defines the maximum packet length in bytes. |

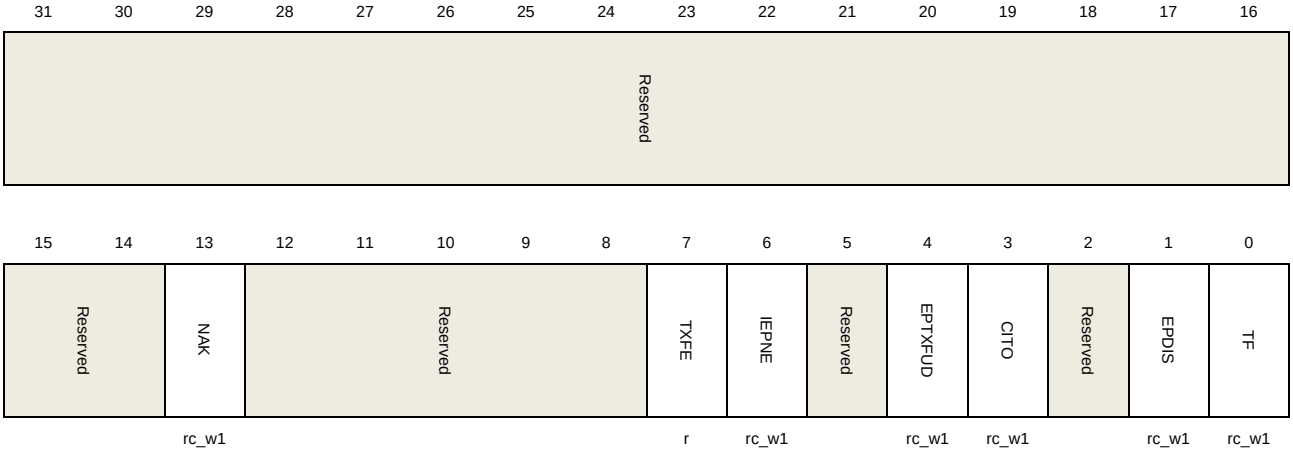
### Device IN endpoint-x interrupt flag register (USBHS_DIEPxINTF) (x = 0..7, where x = endpoint_number)

Address offset: 0x0908 + (endpoint_number × 0x20)

Reset value: 0x0000 0080

This register contains the status and events of an IN endpoint, when software gets an IN endpoint interrupt, it should read this register for the respective endpoint to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1 except the read-only TXFE bit.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:14 | Reserved | Must be kept at reset value. |
| 13 | NAK | NAK handshake sent by USBHS<br>USBHS sets this bit after it sends out a NAK handshake because the NAKS bit in USBHS_DIEPxCTL register is set, or there is no packet data in endpoint's Tx FIFO. |
| 12:8 | Reserved | Must be kept at reset value. |
| 7 | TXFE | Transmit FIFO empty<br>The Tx FIFO of this IN endpoint has reached the empty threshold value defined by TXFTH field in USBHS_GAHBCS register. |
| 6 | IEPNE | IN endpoint NAK effective<br>The setting of SNAK bit in USBHS_DIEPxCTL register takes effect. This bit can be cleared either by writing 1 to it or by setting CNAK bit in USBHS_DIEPxCTL register. |
| 5 | Reserved | Must be kept at reset value. |
| 4 | EPTXFUD | Endpoint Tx FIFO underrun<br>This flag is triggered if the Tx FIFO has no packet data when an IN token is incoming |
| 3 | CITO | Control In Timeout interrupt<br>This flag is triggered if the device waiting for a handshake is timeout in a control IN transaction. |
| 2 | Reserved | Must be kept at reset value. |
| 1 | EPDIS | Endpoint disabled<br>This flag is triggered when an endpoint is disabled from the software's request. |

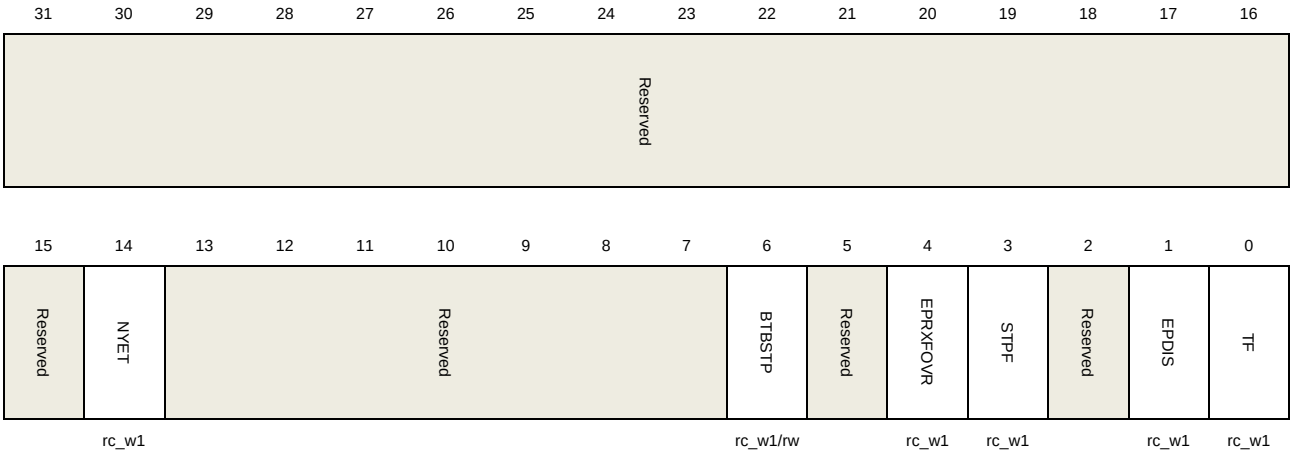
| | | |
|---|---|---|
| 0 | TF | Transfer finished<br>This flag is triggered when all the IN transactions assigned to this endpoint have finished. |

### Device OUT endpoint-x interrupt flag register (USBHS_DOEPxINTF) (x = 0..7, where x = endpoint_number)

Address offset: 0x0B08 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register contains the status and events of an OUT endpoint, when software gets an OUT endpoint interrupt, it should read this register for the respective endpoint to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1.

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:15 | Reserved | Must be kept at reset value. |
| 14 | NYET | NYET handshake is sent<br>This flag is triggered if a NYET handshake is sent by USBHS. |
| 13:7 | Reserved | Must be kept at reset value. |
| 6 | BTBSTP | Back-to-back SETUP packets ( Only for control OUT endpoint)<br>This flag is triggered when a control out endpoint has received more than 3 back-to-back setup packets. |
| 5 | Reserved | Must be kept at reset value. |
| 4 | EPRXFOVR | Endpoint Rx FIFO overrun<br>This flag is triggered if the OUT endpoint's Rx FIFO has no enough space for a packet data when an OUT token is incoming. USBHS will drop the incoming OUT data packet and send a NAK handshake in this case. |
| 3 | STPF | SETUP phase finished (Only for control OUT endpoint) |

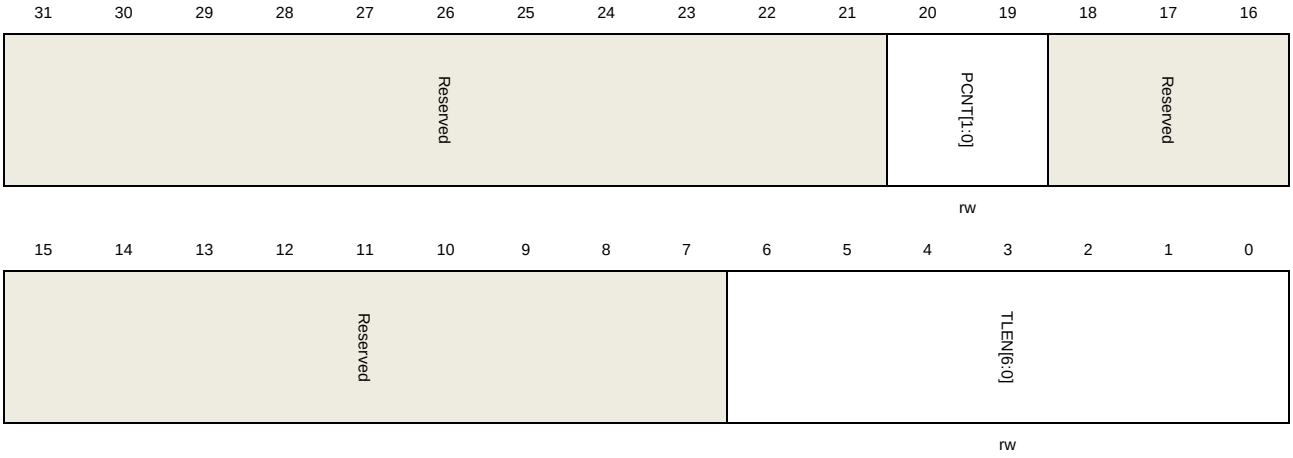
| | | |
|---|---|---|
| | | This flag is triggered when a setup phase finished, i.e. USBHS receives an IN or OUT token after a setup token. |
| 2 | Reserved | Must be kept at reset value. |
| 1 | EPDIS | Endpoint disabled<br>This flag is triggered when an endpoint is disabled from the software's request. |
| 0 | TF | Transfer finished<br>This flag is triggered when all the OUT transactions assigned to this endpoint have finished. |

### Device IN endpoint 0 transfer length register (USBHS_DIEP0LEN)

Address offset: 0x0910

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)



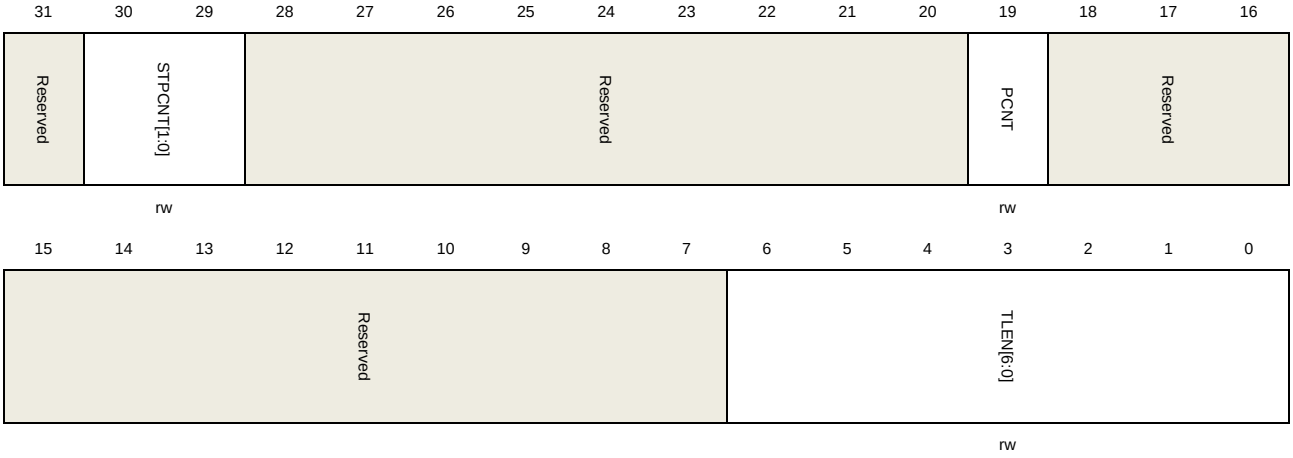

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value. |
| 20:19 | PCNT[1:0] | Packet count<br>The number of data packets desired to be transmitted in a transfer.<br>Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet transmission. |
| 18:7 | Reserved | Must be kept at reset value. |
| 6:0 | TLEN[6:0] | Transfer length<br>The total data bytes number of a transfer.<br>This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Software should program this field before the endpoint is enabled. When software or DMA successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet. |

### Device OUT endpoint 0 transfer length register (USBHS_DOEP0LEN)

Address offset: 0x0B10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:29 | STPCNT[1:0] | SETUP packet count<br>This field defines the maximum number of back-to-back SETUP packets what this endpoint can accept.<br>Software should program this field before setup transfers. Each time a back-to-back setup packet is received, USBHS decreases this field by one. When this field reaches zero, the BTBSTP flag in USBHS_DOEP0INTF register will be triggered.<br>00: 0 packet<br>01: 1 packet<br>10: 2 packets<br>11: 3 packets |
| 28:20 | Reserved | Must be kept at reset value. |
| 19 | PCNT | Packet count<br>The number of data packets is desired to receive in a transfer.<br>Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet reception on bus. |
| 18:7 | Reserved | Must be kept at reset value. |
| 6:0 | TLEN[6:0] | Transfer length<br>The total data bytes number of a transfer.<br>This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Software should program this field before the endpoint is enabled. Each time software or DMA reads out a packet from the Rx FIFO, this field is decreased |

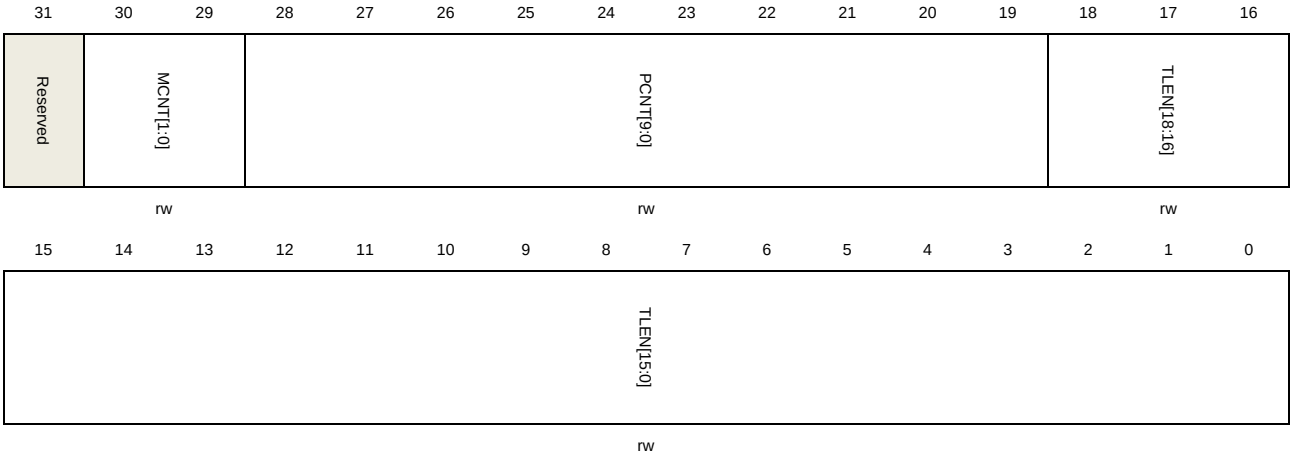
by the byte size of the packet.

### Device IN endpoint-x transfer length register (USBHS_DIEPxLEN) (x = 1..7, where x = endpoint_number)

Address offset: 0x910 + (endpoint_number × 0x20)

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

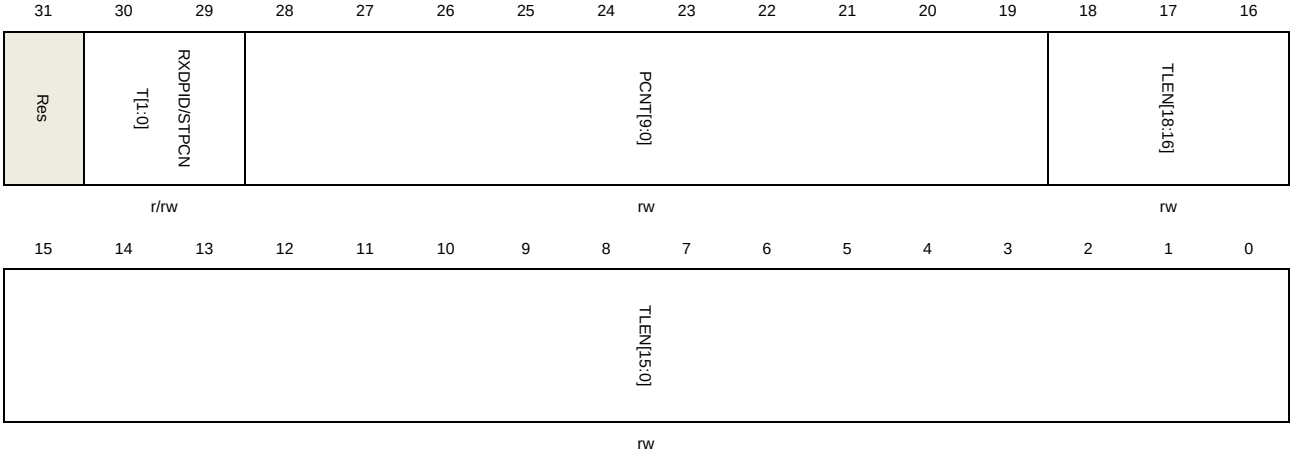



| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:29 | MCNT[1:0] | Multi count<br>This field indicates the number of packets which should be transmitted in a frame<br>01: 1 packet<br>10: 2 packets<br>11: 3 packets |
| 28:19 | PCNT[9:0] | Packet count<br>The number of data packets desired to be transmitted in a transfer.<br>Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet transmission. |
| 18:0 | TLEN[18:0] | Transfer length<br>The total data bytes number of a transfer.<br>This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Software should program this field before the endpoint is enabled. When software or DMA successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet. |

### Device OUT endpoint-x transfer length register (USBHS_DOEPxLEN) (x = 1..7, where x = endpoint_number)

Address offset: 0x0B10 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | Reserved | Must be kept at reset value. |
| 30:29 | RXDPID[1:0] | Received data PID (For isochronous OUT endpoints)<br>This field saves the PID of the latest received data packet on this endpoint.<br>00: DATA0<br>01: DATA2<br>10: DATA1<br>11: MDATA |
| | STPCNT[1:0] | SETUP packet count (For control OUT Endpoints.)<br>This field defines the maximum number back-to-back SETUP packets this endpoint can accept.<br>Software should program this field before setup transfers. Each time a back-to-back setup packet is received, USBHS decreases this field by one. When this field reaches zero, the BTBSTP flag in USBHS_DOEPxINTF register will be triggered.<br>00: 0 packet<br>01: 1 packet<br>10: 2 packets<br>11: 3 packets |
| 28:19 | PCNT[9:0] | Packet count<br>The number of data packets desired to receive in a transfer.<br>Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet reception on bus. |
| 18:0 | TLEN[18:0] | Transfer length |

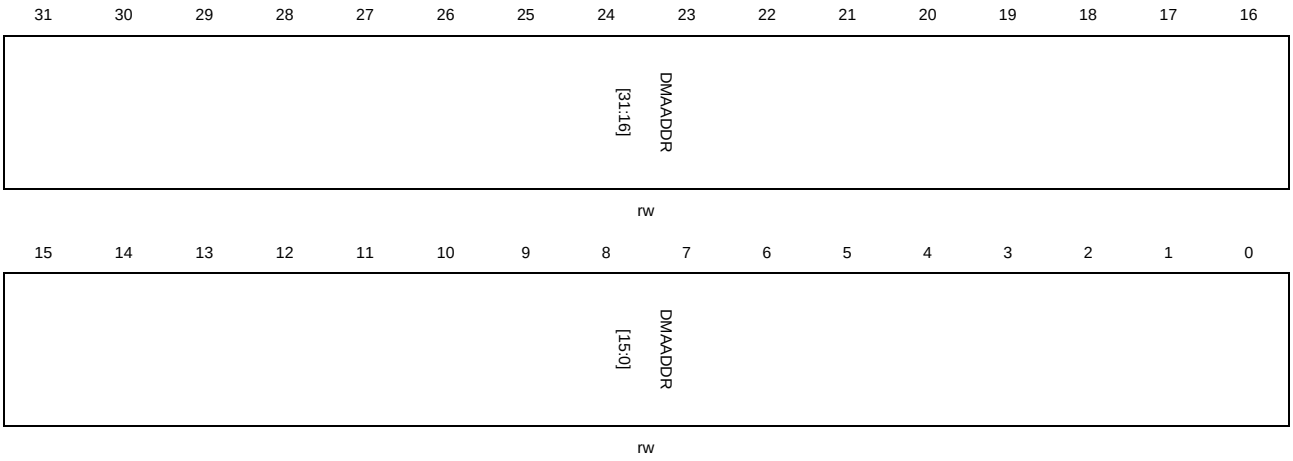
The total data bytes number of a transfer.

This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Software should program this field before the endpoint is enabled. Each time software or DMA reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

### Device IN endpoint-x DMA address register (USBHS_DIEPxDMAADDR) / Device OUT endpoint-x DMA address register (USBHS_DOEPxDMAADDR) (x = 0..7, where x = endpoint_number)

Address offset:
IN endpoint: 0x0914 + (endpoint_number × 0x20)
OUT endpoint: 0x0B14 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DMAADDR[31:0] | DMA address<br>This field defines the endpoint's DMA address. DMA uses this address to fetch packet data for IN endpoint or write packet data for OUT endpoint. |

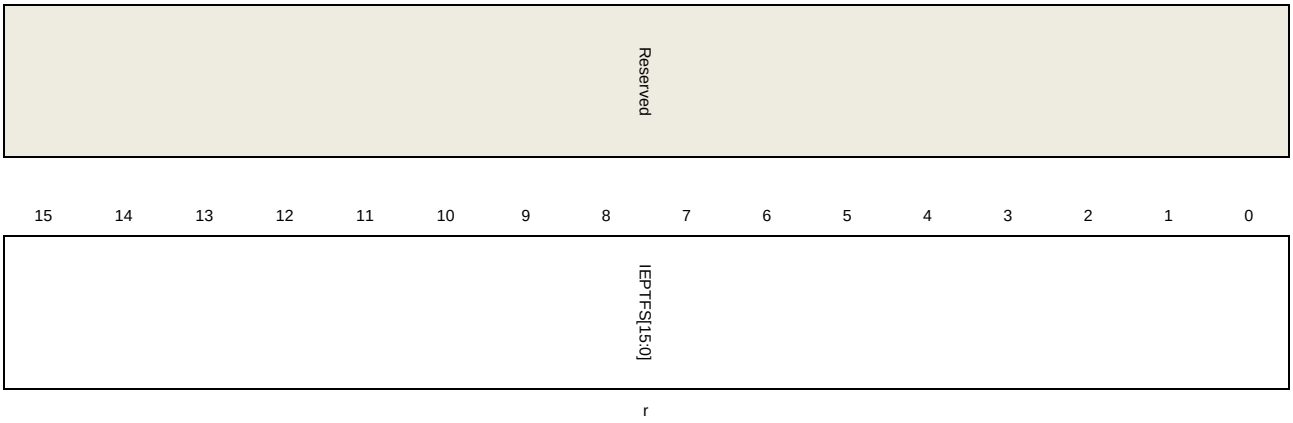
### Device IN endpoint-x transmit FIFO status register (USBHS_DIEPxTFSTAT) (x = 0..7, where x = endpoint_number)

Address offset: 0x0918 + (endpoint_number × 0x20)
Reset value: 0x0000 0200

This register contains the information of each endpoint's Tx FIFO.

This register has to be accessed by word (32-bit)

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16



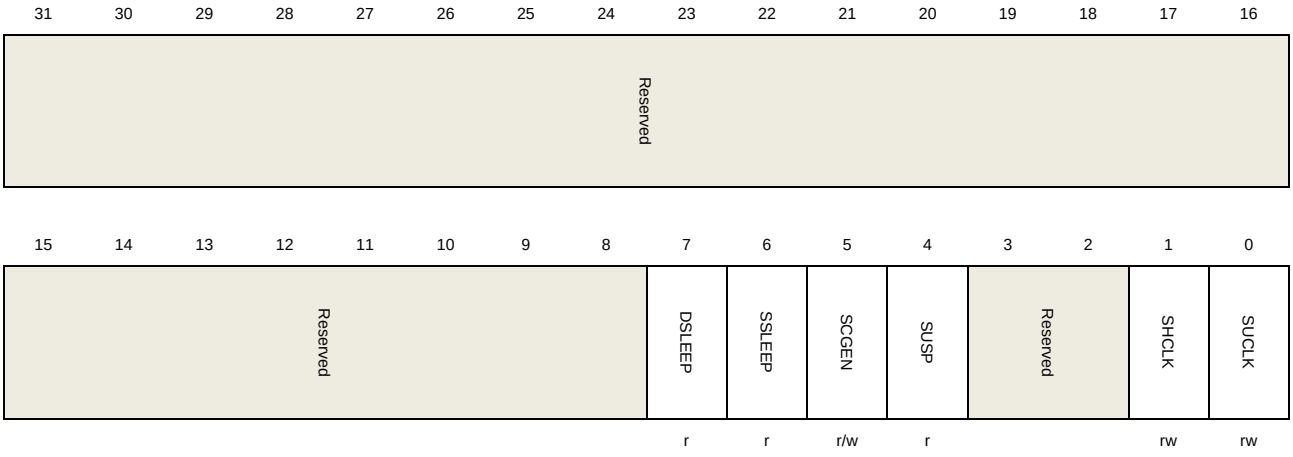

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value. |
| 15:0 | IEPTFS[15:0] | IN endpoint's Tx FIFO space remaining<br>IN endpoint's Tx FIFO space remaining in 32-bit word:<br>0: FIFO is full<br>1: 1 word available<br>…<br>n: n words available |

### 49.7.4. Power and clock control register (USBHS_PWRCLKCTL)

Address offset: 0x0E00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value. |
| 7 | DSLEEP | PHY is in deep sleep status |
| 6 | SSLEEP | PHY is in shallow sleep status |

| | | |
|---|---|---|
| 5 | SCGEN | When this bit is set, the internal clock gating is enabled. |
| 4 | SUSP | PHY is in suspend status |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | SHCLK | Stop HCLK<br>Stop the HCLK to save power.<br>0: HCLK is not stopped<br>1: HCLK is stopped |
| 0 | SUCLK | Stop the USB clock<br>Stop the USB clock to save power.<br>0: USB clock is not stopped<br>1: UCB clock is stopped |

# 50. EtherCAT® SubDevice Controller (ESC) (For GD32H75E)

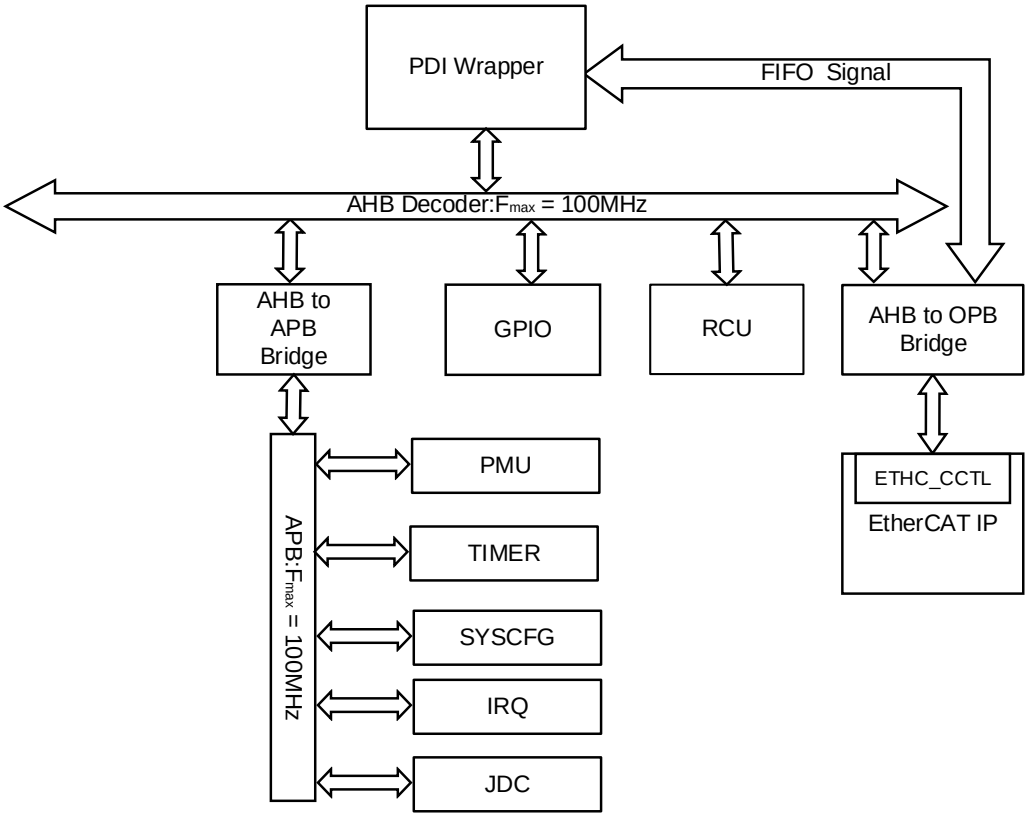
## 50.1. System and bus architecture

### 50.1.1. Bus architecture

The bus architecture of ESC is shown in the following figure. The AHB matrix based on AMBA 5 AHB-LITE is a multi-layer AHB, which enables parallel access paths between one masters and multiple subs in the system. One masters on the AHB decoder, including AHB bus of the PDI Wrapper. The AHB decoder consists of five subs, including the AHB to OPB Bridge, AHB to APB Bridge, GPIO and RCU.

The AHB connects with the AHB peripherals including one AHB-to-APB bridges which provide full synchronous connections between the AHB decoder and the one APB buses. The one APB buses connect with all the APB peripherals, including the PMU, TIMER, SYSCFG, IRQ.

**Figure 50-1. Bus architecture of ESC**




### 50.1.2. Memory map

The following figure shows the memory map of ESC, including registers region and PRAM

region. Each peripheral has several base registers. The memory map summarizes the individual register addresses, whose functions are explained in detail in subsequent chapters.

**Table 50-1. Memory map**

| Pre-defined Regions | ADDRESS | Registers |
|---|---|---|
| EtherCAT | 0x0000 | Type |
| | 0x0001 | Revision |
| | 0x0002 - 0x0003 | Build |
| | 0x0004 | FMMUs Supported |
| | 0x0005 | SyncManagers Supported |
| | 0x0006 | RAM Size |
| | 0x0007 | Port Descriptor |
| | 0x0008 - 0x0009 | ESC Features Supported |
| | 0x0010 - 0x0011 | Configured Station Address |
| | 0x0012 - 0x0013 | Configured Station Alias |
| | 0x0013 - 0x001F | Reserved |
| | 0x0020 | Write Register Enable |
| | 0x0021 | Write Register Protection |
| | 0x0022 - 0x002F | Reserved |
| | 0x0030 | ESC Write Enable |
| | 0x0031 | ESC Write Protection |
| | 0x0032 - 0x003F | Reserved |
| | 0x0040 | ESC Reset ECAT |
| | 0x0041 | ESC Reset PDI |
| | 0x0042 - 0x0FF | Reserved |
| | 0x0100 - 0x0103 | ESC DL Control |
| | 0x0104 - 0x0107 | Reserved |
| | 0x0108 - 0x0109 | Physical Read/Write Offset |
| | 0x0110 - 0x0111 | ESC DL Status |
| | 0x0112 - 0x011F | Reserved |
| | 0x0120 - 0x0121 | AL Control |
| | 0x0122 - 0x012F | Reserved |
| | 0x0130 - 0x0131 | AL Status |
| | 0x0132 - 0x0133 | Reserved |
| | 0x0134 - 0x0135 | AL Status Code |
| | 0x0136 - 0x0137 | Reserved |
| | 0x0138 | RUN LED Override |
| | 0x0139 | Reserved |
| | 0x0140 | PDI Control |
| | 0x0141 | ESC Configuration |
| | 0x0142 - 0x0143 | ASIC Configuration |
| | 0x0144 - 0x0145 | RESERVED Register |

| Pre-defined Regions | ADDRESS | Registers |
|---|---|---|
| | 0x0146 - 0x014F | Reserved |
| | 0x0150 | PDI Configuration |
| | 0x0151 | Sync/Latch PDI Configuration |
| | 0x0152 - 0x0153 | Extended PDI Configuration |
| | 0x0154 - 0x01FF | Reserved |
| | 0x0200 - 0x0201 | ECAT Event Mask |
| | 0x0202 - 0x0203 | Reserved |
| | 0x0204 - 0x0207 | PDI AL Event Mask |
| | 0x0208 - 0x0209 | Reserved |
| | 0x0210 - 0x0211 | ECAT Event Request |
| | 0x0212 - 0x021F | Reserved |
| | 0x0220 - 0x0223 | AL Event Request |
| | 0x0223 - 0x022F | Reserved |
| | 0x0300 - 0x0307 | RX Error Counter |
| | 0x0308 - 0x030B | Forwarded RX Error Counter |
| | 0x030C | ECAT Processing Unit Error Counter |
| | 0x030D | PDI Error Counter |
| | 0x030E | PDI Error Code |
| | 0x030F | Reserved |
| | 0x0310 - 0x0313 | Lost Link Counter |
| | 0x0314 - 0x03FF | Reserved |
| | 0x0400 - 0x0401 | Watchdog Divider |
| | 0x0410 - 0x0411 | Watchdog Time PDI |
| | 0x0420 - 0x0421 | Watchdog Time Process Data |
| | 0x0440 - 0x0441 | Watchdog Status Process Data |
| | 0x0442 | Watchdog Counter Process Data |
| | 0x0443 | Watchdog Counter PDI |
| | 0x0444 - 0x04FF | Reserved |
| | 0x0500 | EEPROM Configuration |
| | 0x0501 | EEPROM PDI Access State |
| | 0x0502 - 0x0503 | EEPROM Control/Status |
| | 0x0504 - 0x0507 | EEPROM Address |
| | 0x0508 - 0x050B | EEPROM Data |
| | 0x050C- 0x050F | Reserved |
| | 0x0510 - 0x0511 | MII Management Control/Status |
| | 0x0512 | PHY Address |
| | 0x0513 | PHY Register Address |
| | 0x0514 - 0x0515 | PHY DATA |
| | 0x0516 | MII Management ECAT Access State |
| | 0x0517 | MII Management PDI Access State |
| | 0x0518 - 0x051B | PHY Port Status |

| Pre-defined Regions | ADDRESS | Registers |
|---|---|---|
| | 0x051C - 0x05FF | Reserved |
| | 0x0600 - 0x06FF | FMMU |
| | 0x0700 - 0x07FF | Reserved |
| | 0x0800 - 0x087F | SyncManager |
| | 0x0880 - 0x08FF | Reserved |
| | 0x0900 - 0x09FF | Distributed Clocks (DC) |
| | 0x0A00 - 0x0AFF | Reserved |
| | 0x0E00 - 0x0E07 | Product ID |
| | 0x0E08 - 0x0E0F | Vendor ID |
| | 0x0E10 - 0x0EFF | Reserved |
| | 0x0F00 - 0x0F03 | Digital I/O Output Data |
| | 0x0F04 - 0x0F0F | Reserved |
| | 0x0F10 - 0x0F17 | General Purpose Outputs |
| | 0x0F18 - 0x0F1F | General Purpose Inputs |
| | 0x0F20 - 0x0F7F | Reserved |
| | 0x0F80 - 0x0FFF | User RAM |
| | 0x1000 - 0x2FFF | Process Data RAM |
| Peripheral | 0x3300 - 0x33FF | AHB2OPB Bridge |
| | 0x3400 - 0x34FF | RCU |
| | 0x3500 - 0x35FF | GPIO |
| | 0x3700 - 0x37FF | PMU |
| | 0x3800 - 0x38FF | TIMER |
| | 0x3900 - 0x39FF | SYSCFG |
| | 0x3A00 - 0x3AFF | IRQ |

### 50.1.3. AHB direct/indirect access

ESC can directly/indirectly access ESC registers and core PRAM through the AHB to OPB bridge in three ways: directly accessible ESC registers, indirectly accessible ESC core registers, and indirectly accessible ESC core PRAM. The bridge provides an AHB Slave interface towards the upstream side and OPB interfaces towards the downstream side of the ESC core, where the upstream AHB side is faster than, in frequency to, the downstream ESC core, and the clocks are synchronous in phase, and have an N:1(max N=16) frequency ratio.

#### Direct AHB transmit access ESC register

Direct access ESC registers are used to transferring data/commands to indirect access ESC core registers. When the AHB bus access address range is 0x0000-0x0FFF, a single register read operation of the ESC core is initiated. When the read period starts, CCTL_BUSY is set to 1, and when the read period ends, CCTL_BUSY is set to 0. The valid data on the AHB bus can be read. The valid data is aligned with the low bits on the bus. When the AHB bus access address range is 0x0000-0x0FFF, a single register write operation of the ESC core is initiated.

The write cycle begins and CCTL_BUSY is set. The write cycle ends and CCTL_BUSY is set to zero. The valid data in the register is written and the valid data is aligned with the low bits on the bus.

### Indirect transmit access ESC core register

ESC can access the ESC core register indirectly through the ESC CCTL_DATA and ESC_CCTL_CMD registers. When reading the ESC core register; write the ESC CCTL command register (CCTL_CMD) once; set ESC CCTL Busy (CCTL_BUSY) bit to 1; ESC CCTL address (CCTL_ADDR) field to the desired register address; set read/write (CCTL_RW) to 1; and set ESC CCTL size (CCTL_SIZE) field to the desired size. When the CCTL_BUSY bit is cleared, valid data can be read, that is, data can be read from the ESC CCTL data register (ESC_CCTL_DATA).

When writing to the ESC core register, write the ESC CCTL command register (ESC_CCTL_CMD) one time. Set the ESC CCTL Busy (CCTL_BUSY) to 1; set the ESC CCTL address (CCTL_ADDR) field to the desired register address; clear the read/write (CCTL_RW) bit, and set the ESC CCTL size (CCTL_SIZE) field to the desired size. Completion of the write cycle is indicated by the ESC CCTL busy (CCTL_BUSY) bit being cleared to zero.

In the above read/write operations, the valid data is always aligned with the low bits of the ESC CCTL data register (CCTL_DATA). The valid data can be referred to the following table.

**Table 50-2. Alignment of valid data**

| CCTL_SIZE | ESC CCTL_ADDR[1:0] | ESC CCTL_DATA valid bytes |
|---|---|---|
| 1 | 00/01/10/11 | [7:0]/ [15:8]/ [23:16]/ [31:24]/ |
| 2 | 00/10 | [15:0]/[31:16] |
| 4 | 00 | [31:0] |

### Indirect transmit access ESC core PRAM

When initiating read operation to core PRAM through AHB, after writing PRAM start address and read length to ESC_PRAM_ALR register, PRAM_BUSY_READ is set to 1, the module starts to initiate multiple OPB read operations, read data from core PRAM and write to the TX FIFO. All OPB read operations are complete. PRAM_BUSY_READ is cleared. When data is transferred from the ESC core to the TX FIFO, the PRAM read length PRAM_LEN_READ and the PRAM read address PRAM_ADDR_READ are updated to show the process. Determines the valid bytes of the first read data according to the start address. Determines the valid bytes of the last read data based on the starting address and operation length. If necessary, the read command can be stopped by setting the ESC_PRAM_CR[PRAM_STOP_READ] bit to 1. If the OPB read period starts, the stop command takes effect after the current read operation is complete. After the stop command takes effect, data in the TX FIFO is cleared.

When writing to the PRAM through AHB, after writing the RAM start address and write length to the ESC_PRAM_ALW register, write 1 to the PRAM_BUSY_WRITE bit of the

ESC_PRAM_CW register, the module initiates multiple OPB write operations. Data is read from the RX FIFO and written to the core PRAM. All OPB write operations are complete. PRAM_BUSY_WRITE is cleared Write operations support wait mechanism, PRAM_BUSY_WRITE bit is set to 1, but when RX FIFO is empty, OPB module will not immediately initiate OPB transfer operation, until there is data in RX FIFO will initiate this operation. After each OPB ACK response, it will detect the RX FIFO state. If the RX FIFO is empty, it will enter the waiting state and enter the next transmission operation until there is data. When data is transferred from the RX FIFO to the core, the PRAM write length PRAM_LEN_WRITE and the PRAM write address PRAM_ADDR_WRITE are updated to show the process. Determines the valid bytes of the first data write according to the start address. Based on the starting address and operation length, determines the valid bytes of the last data write.

If necessary, the write command can be stopped by setting the PRAM_STOP_WRITE bit to 1. If the OPB write cycle starts, the stop command takes effect after the read operation is complete. After the stop command takes effect, data in the RX FIFO is cleared.

### 50.1.4. Register protection in BUSY state

If the BUSY related register bits, such as CCTL_BUSY, PRAM_BUSY_READ, and PRAM_BUSY_WRITE, are set to 1, you can set the BRP bit to protect the register from rewriting. When CCTL_BUSY is set to 1, ESC_CCTL_DATA/ ESC_CCTL_CMD is protected from rewriting by AHB write operations. When PRAM_BUSY_READ is set to 1, ESC_PRAM_ALR is protected from rewriting by AHB write operations. When PRAM_BUSY_WRITE is set to 1, ESC_PRAM_ALW is protected from being overwritten by AHB write operations

When the BRP bit is set to 1, when the BUSY related register bit is set to 1, the user's AHB write operation to the corresponding register will be lost, then the WDLF flag in the ESC_OPB_CS register is set to 1, and the interrupt will be triggered when the WDIE bit is set to 1.

When the BRP bit is set to 0, when the BUSY related register bit is set to 1, the user's AHB write operation to the corresponding register will cause the current OPB transmission error, then the WEF flag in the ESC_OPB_CS register is set to 1, and the interrupt will be triggered when the WEIE bit is set to 1. It is advised to handle it as soon as possible to avoid more errors.

### 50.1.5. OPB transmission timeout function

The OPB transmission timeout function can be enabled by setting the TOEN bit in the ESC_OPB_CS register to 1. The timeout interval can be configured by the TO_CNT bit in the ESC_OPB_CS register. When the counter exceeds the TO_CNT programming value, the TOF flag in the ESC_OPB_CS register is set to 1. If the TOIE bit in the ESC_OPB_CS register is set to 1, a timeout interrupt is generated. Timeout interrupt response processing:

1. If the ESC CCTL direct read/write mode is used, the TOF flag is set after a timeout interrupt response occurs. Terminate the transmission directly to avoid the BUS being occupied.

2. If the ESC CCTL is used in indirect read/write mode, the TOF flag is set after a timeout interrupt response occurs. You can stop this operation by writing 1 to the ESC CCTL_STOP bit.

3. When the PRAM is used in indirect read/write mode, the TOF flag is set after a timeout interrupt response occurs. You can stop this operation by writing 1 to the PRAM_STOP_WRITE/ PRAM_STOP_READ bit.

### 50.1.6. ESC core controller (ESC_CCTL)

The main purposes of the ESC core controller (ESC_CCTL) are the following:

- Configuring indirect transmit access of the ESC core register
- Configuring indirect transmit access of the ESC core PRAM

### 50.1.7. ESC core controller register definition

ESC core control register base address: 0x3300

#### ESC CCTL data register (ESC_CCTL_DATA)

Address offset: 0x00
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCTL_DATA[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCTL_DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CCTL_DATA[31:0] | ESC CCTL data<br>This field indicate the value read from or written to the ESC Core. Reading or writing depends on the ESC_CCTL_CMD[CCTL_RW] bit. If the CCTL_RW bit is 1, this value is the data read from the ESC Core; If the CCTL_RW bit is 0, this value is the data written to the ESC Core.<br>The low bit of this filed always indicates the valid data written or read. |

#### ESC CCTL command register (ESC_CCTL_CMD)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CCTL_BUSY | CCTL_RW | CCTL_STOP | Reserved | | | | | | | | | | CCTL_SIZE[2:0] | | |
| rw | rw | rw | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CCTL_ADDR[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | CCTL_BUSY | CCTL Busy. There is no effect if the bit is written to zero<br>0: No read/write (according to CCTL_RW) operation is being performed.<br>1: Read/write operation is being performed.<br>**Note:** When the read/write operation is complete, this bit will be cleared. Then the valid data is available for the HOST to read from ESC_CCTL_DATA or write to ESC_CCTL_DATA register. It is required that ESC_CCTL_CMD and ESC_CCTL_DATA registers can modified when this bit is 0. |
| 30 | CCTL_RW | Read operation or write operation<br>0: Write operation<br>1: Read operation |
| 29 | CCTL_STOP | Stop read operation or write operation. There is no effect if the bit is written to zero<br>0: No effect<br>1: Stop the ESC core register reading or writing operation<br>**Note:** When the CCTL_BUSY is cleared, the CCTL_STOP will be cleared |
| 28:19 | Reserved | Must be kept at reset value |
| 18:16 | CCTL_SIZE[2:0] | This field specifies the ESC CCTL size (byte), 1, 2 and 4 are valid, other values are invalid. More details refer to ***AHB direct/indirect access***. |
| 15:0 | CCTL_ADDR[15:0] | This field specifies the addresses of ESC core registers that will be accessed |

### ESC PRAM FIFO data read register (ESC_PRAM_FIFO_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRAM_FIFO_DATA_READ[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PRAM_FIFO_DATA_READ[15:0] | | | | | | | | | | | | | | | |

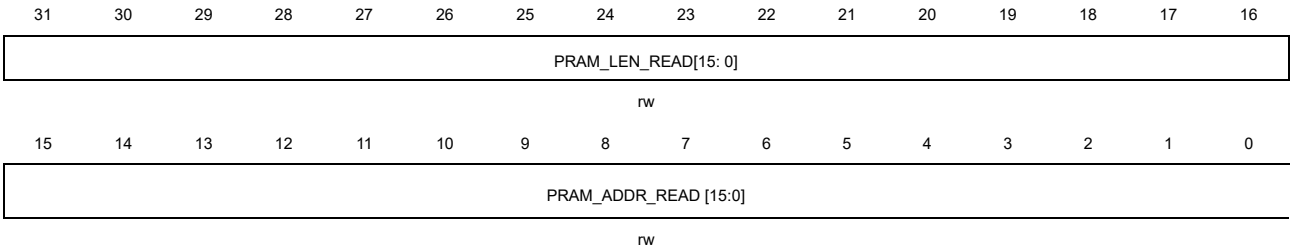
rw

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PRAM_FIFO_DATA_READ[31:0] | Data read from ESC PRAM.<br>The valid value of data is determined according to the start address and the transfer length. |

### ESC PRAM address and length read register (ESC_PRAM_ALR)

Address offset: 0x14
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

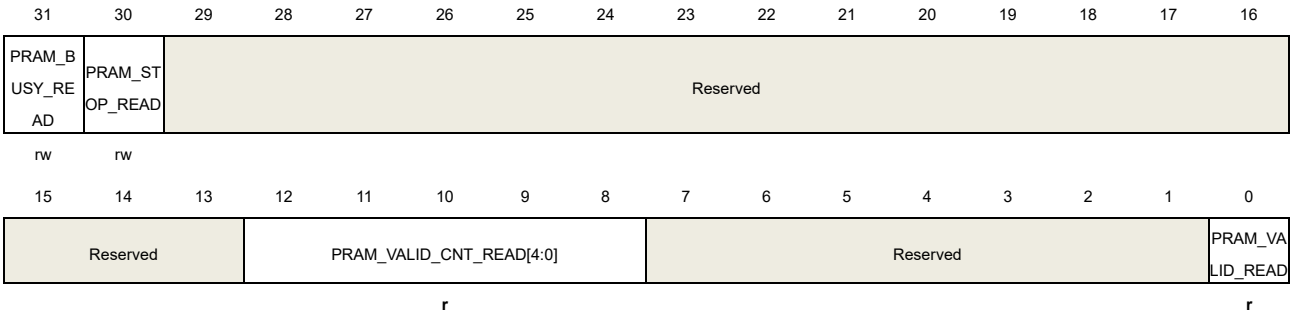



| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | PRAM_LEN_READ[15: 0] | Data length read from ESC PRAM in bytes. This field is decremented as data is read into the FIFO.<br>**Note:** When PRAM_BUSY_READ is 1, this field cannot be modified. |
| 15:0 | PRAM_ADDR_READ [15:0] | ESC PRAM data read address. This field is incremented as data is read into the FIFO.<br>**Note:** When PRAM_BUSY_READ is 1, this field cannot be modified. |

### ESC PRAM command read register (ESC_PRAM_CR)

Address offset: 0x18
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31 | PRAM_BUSY_READ | Whether the PRAM is being read. There is no effect if the bit is written to zero<br>0: No PRAM read operation<br>1: PRAM is being read<br>**Note:** This bit is cleared when the read operation is complete. |
| 30 | PRAM_STOP_READ | Stop PRAM read operation. There is no effect if the bit is written to zero<br>0: No effect<br>1: Stop the PRAM reading operation<br>Note: After this bit is set to 1, the PRAM_BUSY_READ is cleared, and the RX FIFO is reset. Then this bit will self-clear. |
| 29:13 | Reserved | Must be kept at reset value |
| 12:8 | PRAM_DATA_CNT_READ[4:0] | PRAM data read valid count<br>When data is read from PRAM to RX FIFO, this count increases. When the MCU reads RX FIFO, this count decreases. |
| 7:1 | Reserved | Must be kept at reset value |
| 0 | PRAM_VALID_DATA_READ | PRAM valid data read<br>0: There is no valid data to be read.<br>1: There is valid data to be read. |

### ESC PRAM FIFO data write register (ESC_PRAM_FIFO_DW)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRAM_FIFO_DATA_WRITE[31:16] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRAM_FIFO_DATA_WRITE[15:0] | | | | | | | | | | | | | | | |
| **rw** | | | | | | | | | | | | | | | |

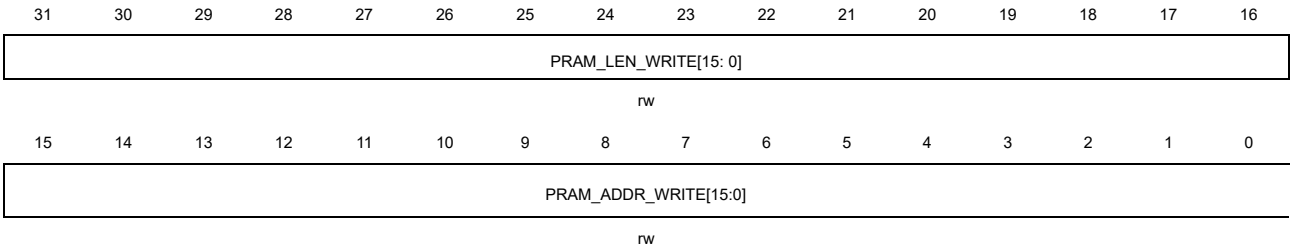
| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PRAM_FIFO_DATA_WRITE[31:0] | Data write to ESC PRAM.<br>The valid value of data is determined according to the start address and the transfer length. |

### ESC PRAM address and length write register (ESC_PRAM_ALW)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | PRAM_LEN_WRITE[15: 0] | Data length written to ESC PRAM in bytes. This field is decremented as data is written into the FIFO.<br>**Note:** When PRAM_BUSY_WRITE is 1, this field cannot be modified. |
| 15:0 | PRAM_ADDR_WRITE[15:0] | ESC PRAM data write address. This field is incremented as data is read from the FIFO.<br>**Note:** When PRAM_BUSY_ WRITE is 1, this field cannot be modified. |

### ESC PRAM command write register (ESC_PRAM_CW)

Address offset: 0x28

Reset value: 0x0000 1001

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31 | PRAM_BUSY_WRITE | Whether the PRAM is being written. There is no effect if the bit is written to zero<br>0: No PRAM write operation<br>1: PRAM is being writing<br>**Note:** This bit is cleared when the write operation is complete. |
| 30 | PRAM_STOP_WRITE | Stop PRAM write operation. There is no effect if the bit is written to zero<br>0: No effect<br>1: Stop the PRAM writing operation<br>Note: After this bit is set to 1, the PRAM_BUSY_WRITE is cleared, and the TX FIFO is reset. Then this bit will self-clear. |
| 29:13 | Reserved | Must be kept at reset value |

| | | |
|---|---|---|
| 12:8 | PRAM_VALID_CNT_WRITE[4:0] | PRAM data write valid count<br>When data is written to TXFIFO by the MCU, this count decreases. When TXFIFO writes to PRAM, this count increases. |
| 7:1 | Reserved | Must be kept at reset value |
| 0 | PRAM_VALID_DATA_WRITE | PRAM valid data write<br>0: There is no valid data to be written<br>1: There is valid data to be written. |

### ESC OPB control and status register (ESC_OPB_CS)

Address offset: 0x30

Reset value: 0x0000 04A0

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAAF | RAAIE | WDLF | WDLIE | TOF | TOIE | WEF | WEIE | ESC CCTLIVF | ESC CCTLIVIE | Reserved | | | | | |
| rc_w1 | rw | rc_w1 | rw | rc_w1 | rw | rc_w1 | rw | rc_w1 | rw | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | BRP | TO_CNT[8:0] | | | | | | | | | TOEN |
| | | | | | rw | rw | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | RAAF | Reserved address access flag. This flag will be triggered when the access address is a reserved address segment. This flag can be cleared by writing 1.<br>0: The access address is a valid address segment<br>1: The access address is a reserved address segment |
| 30 | RAAIE | Reserved address access interrupt enable<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the RAAF bit is set |
| 29 | WDLF | Write data lost flag. when BRP enable, AHB write operation will cause data lose when the busy bit is high, this flag will be triggered. This flag can be cleared by writing 1.<br>0: No data lost<br>1: The data in this transmission is lost |
| 28 | WDLIE | Write data lost interrupt enable<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the RAAF bit is set |
| 27 | TOF | Time out flag, when single transmission time exceeds the programmed value, this flag will be triggered. This flag can be cleared by writing 1.<br>0: No ESC core timeout transmission occurred |

| | | |
|---|---|---|
| | | 1: An ESC core timeout transmission occurred |
| 26 | TOIE | Timer out interrupt enable<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the RAAF bit is set |
| 25 | WEF | write error flag. when BRP disable, AHB write operation will cause current OPB transmission error when the busy bit is high, this flag will be triggered. This flag can be cleared by writing 1. |
| 24 | WEIE | Write error interrupt enable.<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the WEF bit is set |
| 23 | ESC CCTLIVF | CCTL_SIZE & CCTL_ADDR illegal value flag. The CCTL_SIZE and CCTL_ADDR value written to the register does not meet the requirement, causing this transfer to not occur, this flag will be triggered. This flag can be cleared by writing 1. |
| 22 | ESC CCTLIVIE | CCTL_SIZE & CCTL_ADDR Illegal value interrupt enable.<br>0: Interrupt is inhibited<br>1: An interrupt will occur whenever the IESC CCTLVF bit is set |
| 21:11 | Reserved | Must be kept at reset value. |
| 10 | BRP | When busy bit is high, prevent register change<br>1: the protection takes effect<br>0: the protection does not take effect |
| 9:1 | TO_CNT[8:0] | Time Out counter<br>This field indicates the transmission timeout in bits during which no ACK response was received; The programmer data cannot be less than the minimum value recommended the manual(60). |
| 0 | TOEN | Time Out enable<br>This field indicates that whether the function is effect<br>1: Timeout feature enabled<br>0: Timeout feature disabled. |

### 50.1.8. System configuration controller (ESC_SYSCFG)

The main purposes of the system configuration controller (SYSCFG) are the following:

- Configuring MCU HCLK frequency ratio
- Configuring SPI extend mode
- Providing Chip ID and version

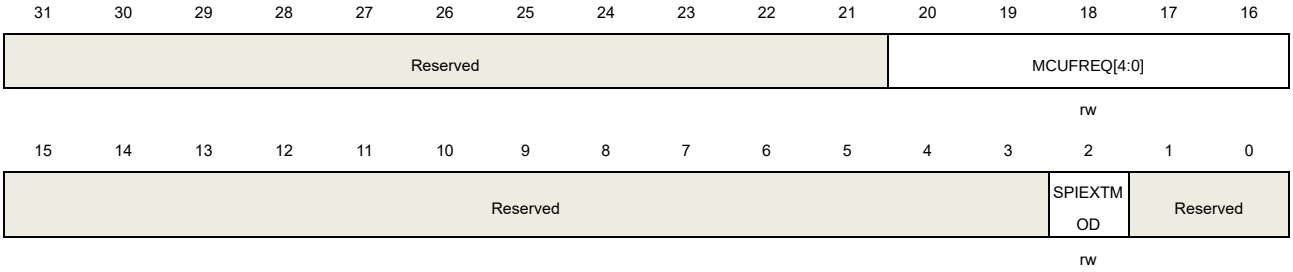
### 50.1.9. System configuration register definition

SYSCFG base address: 0x3900

### System configuration register 0 (ESC_SYSCFG_CFG0)

Address offset: 0x00

Reset value: 0x001F 0000



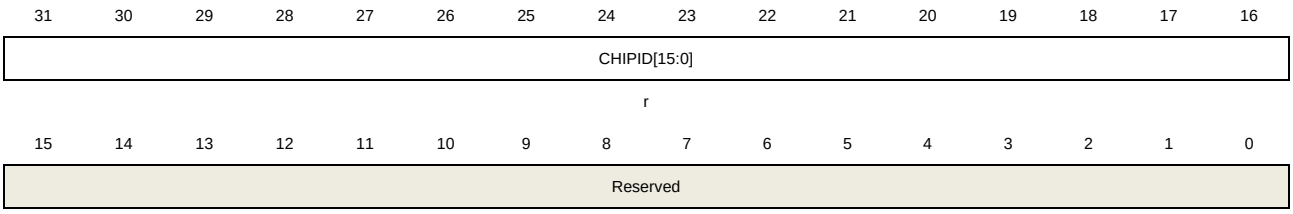

| Bits | Fields | Descriptions |
|---|---|---|
| 31:21 | Reserved | Must be kept at reset value |
| 20:16 | MCUFREQ[4:0] | MCU HCLK frequency ratio<br>00000: MCU_HCLK_FREQ >=100MHZ<br>00001: 100MHZ/2 <= MCU_HCLK_FREQ < 100MHZ<br>00010: 100MHZ/3 <= MCU_HCLK_FREQ < 100MHZ/2<br>…..<br>11111: 100MHZ/32 <= MCU_HCLK_FREQ < 100MHZ/31 |
| 15:8 | Reserved | Must be kept at reset value |
| 7:3 | Reserved | Must be kept at reset value |
| 2 | SPIEXTMOD | PDI type combined with SPI<br>0: GPIO<br>1: MII (OSPI: without clk_25m output; Others: with clk_25M) |
| 1:0 | Reserved | Must be kept at reset value |

### SYSCFG chip id register (ESC_SYSCFG_CHIPID)

Address offset: 0x90

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).



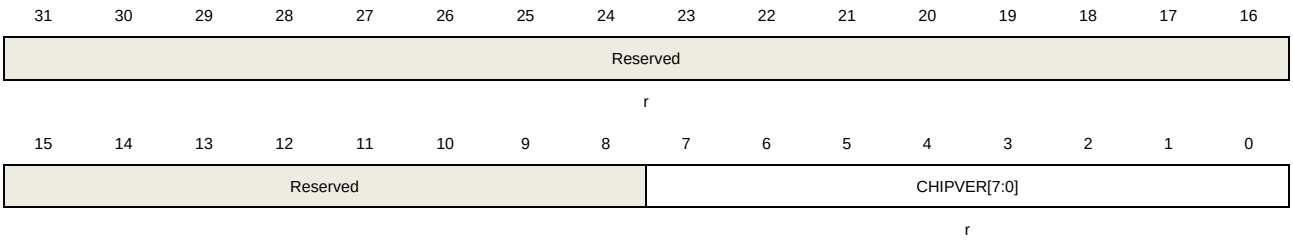

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:16 | CHIPID[15:0] | Chip ID |
| 15:0 | Reserved | Must be kept at reset value |

### SYSCFG chip version register (ESC_SYSCFG_CHIPVER)

Address offset: 0x94
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

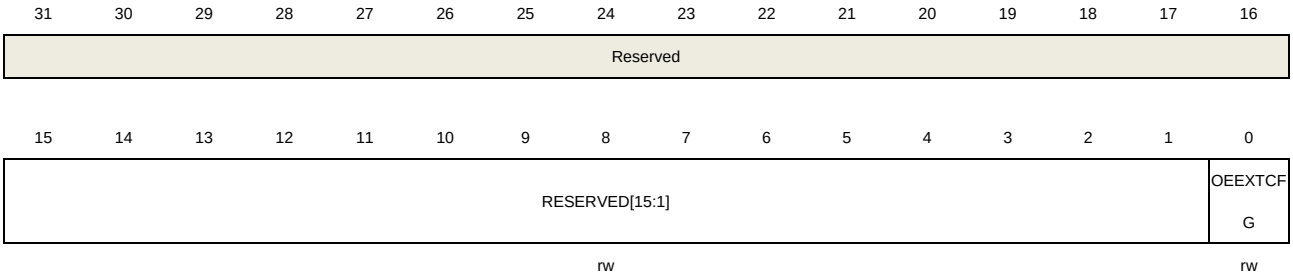



| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | Reserved | Must be kept at reset value |
| 15:0 | CHIPVER[7:0] | Chip version |

### SYSCFG reserved register (ESC_SYSCFG_RESERVED)

Address offset: 0xF0
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:1 | RESERVED[15:1] | Reserved register |
| 0 | OEEXTCFG | OE_EXT output clock configuration bit<br>0: Enable OE_EXT clock output<br>1: Disable OE_EXT clock output<br><br>**Note:** This bit is configured to 0 by default to enable the OE_EXT clock output, which may lead to excessive EMI radiation. When addressing EMI radiation issues, |

set this bit to 1 to disable the OE_EXT clock output.

# 50.2. Power management unit (PMU)

## 50.2.1. Overview

The power consumption is regarded as one of the most important issues for the devices of ESC series. Power management unit (PMU) provides four types of device level and three types of module level power saving modes, device level power saving modes including MOD0, MOD1, MOD2 and MOD3, module level power saving modes including EtherCAT clock management, PHY power management and the LED pins power management. These modes reduce the power consumption and allow the application to achieve the best tradeoff among the conflicting demands of device operating time, speed and power consumption. PMU also supports wake up event detection and power management event (PME) notification.

## 50.2.2. Characteristics

- EtherCAT clock management.
- PHY power management, including PHY A and B energy detect(ED) power down management and common power down management.
- LED pins power down management.
- Four types of device level power saving modes, including MOD0, MOD1, MOD2 and MOD3.
- PHY wake up event detection, including PHY ED powered wake up and PHY LAN magic packet wake up.
- Interrupt wake up notification.

## *50.2.3.* Function overview

### Device ready

The bit RDY in PMU_CTL0 register, can indicate whether the device is ready. The master processor can read this bit to obtain the ready status of the device.

- After power on, EtherCAT device reset or digital reset, if RDY bit is set, it indicates that the device has successfully read the contents of the EEPROM and is configured according to the read contents.
- Setting the ETHERCAT_RST bit in the RESET_CTL register will reset the EtherCAT core, which will cause the EtherCAT core to re-read the EEPROM and reconfigure according to the configuration content. During this process, the RDY bit will momentarily transition to a low level.
- When the device enters power-saving modes MOD1, MOD2, or MOD3, the RDY bit will transition to a low level. Once the device is awakened from a power-saving mode back into the MOD0, and after the PLL becomes stable, the RDY bit will be set back to a high level.

**Note**: The device supports voltage detection. The RDY bit can be set when the supply voltage reaches a predetermined value.

### PHY wake up event detection

The device supports two types of PHY wake-up event detection:

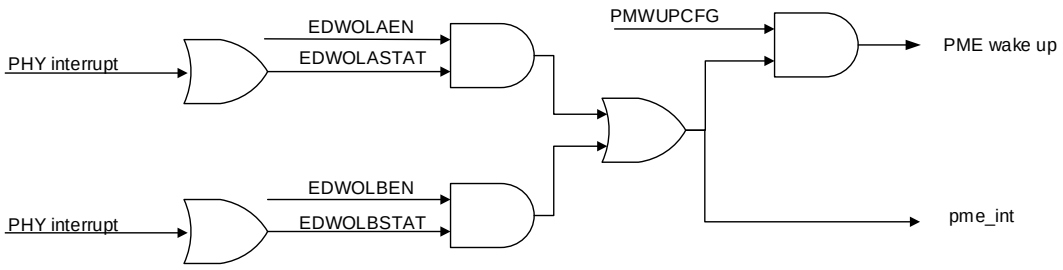
- PHY ED powered wake up event.
- PHY LAN magic packet wake up.

### PME wake up notification

The latches of bit EDWOLASTAT and EDWOLBSTAT in the PMU_CTL register is handled in the PME module. Refer to the ***Figure 50-2. PME interrupt pending*** to understand the logic of PME interrupt control. If EDWOLAEN or EDWOLBEN is set, when energy detect / WoL event happened on port A or B PHY the PME_INT bit in interrupt status register will be set.

When PMWUPCFG is set, PME events can automatically wake up the system in some device-level power saving modes.

**Figure 50-2. PME interrupt pending**




### Module level power saving modes

The device supports three types of module level power saving modes:

- The ECATCLKDIS bit in PMU_CTL0 register can use to disable EtherCAT core clock.
- PHY power management
  - PHY A and B ED power down management, support auto ED power down.
  - Common power down management.
- LEDs output management
  - The LEDOUTDIS bit in PMU_CTL0 register can use to disable LEDs output.
  - The LEDMODCFG bit in PMU_CTL0 register can use to configure LEDs working mode (take effect only when LEDOUTDIS is set).
  - The LEDINACT bit in PMU_CTL0 register can use to configure the inactive state when LEDs work in push-pull mode (take effect only when LEDOUTDIS is set).

### Device level power saving modes

After a device level reset, the device operates at full function and all clocks are active. Users can achieve lower power consumption through gating the clocks of the unused functions. Besides, four device level power saving modes are provided to achieve even lower power consumption, they are MOD0, MOD1, MOD2 and MOD3.

#### MOD0

After a device level reset, the device works in MOD0 and operates at full function, all clocks are active.

#### MOD1

When in MOD1, device will disable all clocks derived from the PLL clock. If powered via PHY or externally, the network clock remains enabled. The XTAL and PLL remain enabled. This mode can be exited either manually or automatically.

This mode is applicable to the PHY's common power down management, PHY's WoL (Wake on LAN) mode, and PHY's ED power down management.

#### MOD2

When in MOD2, device will disable all clocks derived from the PLL clock. If powered via PHY or externally, the network clock remains enabled. It is allowed to disable the PLL (it will be disabled if both PHYs are in ED or common power down management). The XTAL and PLL remain enabled. This mode can be exited either manually or automatically.

This mode is applicable to the PHY's common power down management, PHY's WoL (Wake on LAN) mode, and PHY's ED power down management.

#### MOD3

When in MOD3, device will disable all clocks derived from the PLL clock. The PLL is disabled. The external network clocks are turned off. The crystal oscillator is disabled. This mode can only be exited manually.

This mode is applicable to the PHY's common power down management.

Before setting this power state, the master device should set PHY_PWR_DWN bit.

**Table 50-3. Power saving mode summary**

| Mode | PLL | System clocks | Network clocks | XTAL |
|---|---|---|---|---|
| **MOD0** | ON | ON | usable | ON |
| **MOD1** | ON | OFF | usable | ON |
| **MOD2** | OFF | OFF | usable | ON |
| **MOD3** | OFF | OFF | OFF | OFF |

### Entering device level power saving modes

To transition from MOD0 to MOD1, MOD2, or MOD3, can follow these steps:

1. Configure PMWUPCFG bit.
2. Configure PHY wake up detection, about PHY wake up detection can refer to ***PHY wake up event*** detection.
3. Configure PHY wake up notification, about PHY wake up notification can refer to ***PME wake up notification***.
4. ensure that the device has been able to enter the power saving mode (ensure that there is no need to send data packets, receivers disabled, etc.).
5. Set PMSLPEN bit.

**Note**:

- After entering power saving mode, the RDY bit in registers PMU_CTL0 will be set to low level.
- After entering power saving mode, the master interface is invalid.

### Exiting device level power saving modes

The device level power saving modes can be exited either manually or automatically.

If PMWUPCFG bit is set, the PME wake up is enabled, PME automatically wake up may occur. About PME wake up can refer to ***PME wake up notification***.

The master can manually wake up device by:

- Perform SPI / SQI cycles on the device (SCS# low and SCK high). Although all read and write operations are ignored until the device is woken up, the master should still indicate to wake up the device by reading the PMU_PDIREFVAL register. No attempt should be made to read and write any other address until the device has been woken up.

**Note**:

- The working state of the master interface can be determined by reading the PMU_PDIREFVAL register. Once the correct value is read, the master interface will enter the ready state. Then the RDY bit will indicate when the device is fully awakened.
- After automatic or manual wakeups, the device RDY bit is set once the device has returned to MOD0 and the PLL has been re-stabilized, the PMMODCFG and PMSLPEN bits or bit will be cleared (set to 0).
- If all is well, the device wake up time should be less than 2 ms.

## 50.2.4. Register definition

PMU base address: 0x0000 3700

### Control register 0 (PMU_CTL0)

Address offset: 0x00
Reset value: 0x0000 C000.

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PMMODCFG[1:0] | | Reserved | | PMWUPCFG | LEDOUTDIS | LEDMODCFG | LEDINACT | Reserved | | ECATCLKDIS | Reserved | | | EDWOLBSTAT | EDWOLASTAT |
| rw | | | | rw | rw | rw | rw | | | rw | | | | r_w1 | r_w1 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EDWOLBEN | EDWOLAEN | Reserved | | | | | | | | | | | | | RDY |
| rw | rw | | | | | | | | | | | | | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | PMMODCFG[1:0] | Power management mode configuration bits<br>When PMSLPEN is set, these bits can be used to configure the power management mode These bits are cleared when the device wakes up.<br>00: MOD0<br>01: MOD1<br>10: MOD2<br>11: MOD3 |
| 29:28 | Reserved | Must be kept at reset value |
| 27 | PMWUPCFG | Power management wake up mode configuration bit<br>0: Wake up by master<br>1: Wake up by PME or master |
| 26 | LEDOUTDIS | LEDs output disabled bit<br>When this bit is set, the output of LEDs will be disabled. When work in open-drain / open-source mode LEDs will not be driven, when LEDs work in push-pull mode LEDs will still be driven but the status of LEDs are inactive.<br>0: LEDs output are enabled<br>1: LEDs output are disabled |
| 25 | LEDMODCFG | LEDs working mode configuration bit (take effect only when LEDOUTDIS is set)<br>0: The working mode of LEDs is open-drain / open-source<br>1: The working mode of LEDs is push-pull |
| 24 | LEDINACT | Push-pull mode LEDs inactive state configuration bit (take effect only when LEDOUTDIS is set)<br>0: 0 is inactive state |

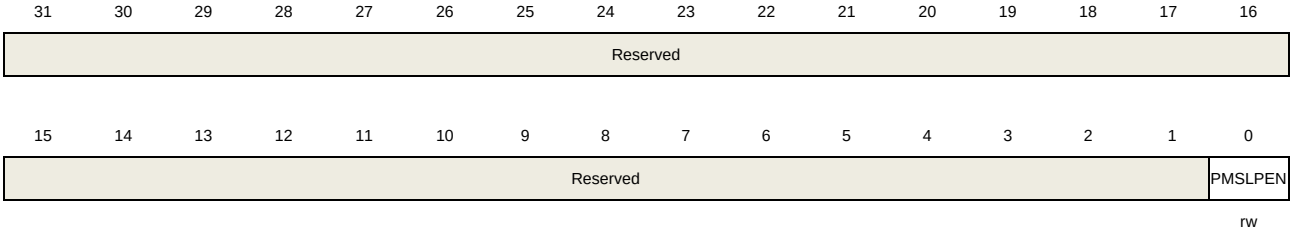
| | | |
|---|---|---|
| | | 1: 1 is inactive state |
| 23:22 | Reserved | Must be kept at reset value |
| 21 | ECATCLKDIS | EtherCAT core clock disable bit<br>EtherCAT core clock will disable if ECATCLKDIS is set. To set this bit need write 1 twice in a row. Writing 0 will clear the count.<br>0: Enable EtherCAT core clock<br>1: Disable EtherCAT core clock |
| 20:18 | Reserved | Must be kept at reset value |
| 17 | EDWOLBSTAT | Energy detect / WoL port B status bit<br>To clear this bit, the events on the PHY need to clear first.<br>0: No energy detect / WoL event happened on port B PHY<br>1: energy detect / WoL event happened on port B PHY |
| 16 | EDWOLASTAT | Energy detect / WoL port A status bit<br>To clear this bit, the events on the PHY need to clear first.<br>0: No energy detect / WoL event happened on port A PHY<br>1: energy detect / WoL event happened on port A PHY |
| 15 | EDWOLBEN | Energy detect / WoL port B enable bit<br>When energy detect / WoL event happened on port B PHY and this bit is enabled, the PMUIF bit in interrupt status register will be set.<br>0: Disable energy detect / WoL port B<br>1: Enable energy detect / WoL port B |
| 14 | EDWOLAEN | Energy detect / WoL port A enable bit<br>When energy detect / WoL event happened on port A PHY and this bit is enabled, the PMUIF bit in interrupt status register will be set.<br>0: Disable energy detect / WoL port A<br>1: Enable energy detect / WoL port A |
| 13:1 | Reserved | Must be kept at reset value |
| 0 | RDY | Device ready bit<br>This bit indicates whether the device is ready. The master processor can read this bit to obtain the ready status of the device, after power on, EtherCAT device reset, module reset, digital reset or leave from power savings mode.<br>0: Device is not ready<br>1: Device is ready<br>**NOTE**:<br>■ The rising edge of this bit will set the READYIF bit in INTC_FLAG register, and can trigger an interrupt.<br>■ When this bit is clear read access to any internal resource is prohibited except for the PMU_CTL, PMU_PDIVAL, and RESET_CTL registers.<br>■ Before this bit is set, write operations to any address are invalid. |

### Control and status register (PMU_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).



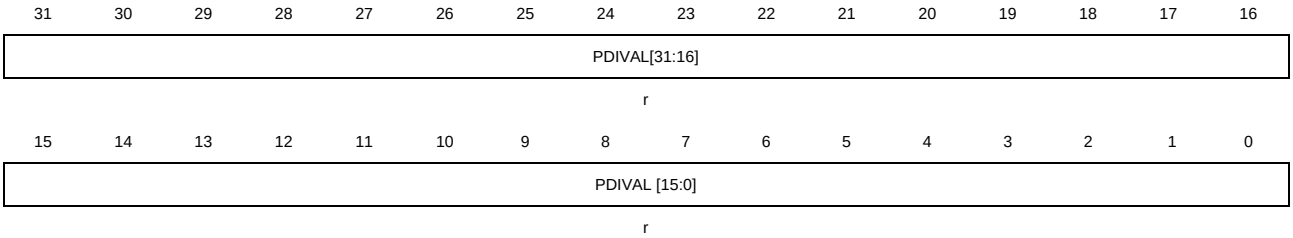

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value |
| 0 | PMSLPEN | Power management sleep mode enable bit<br>When PMSLPEN is set, the device will enter the power management mode which is configured by PMMODCFG bits.<br>0: Disable power management sleep mode<br>1: Enable power management sleep mode<br>**NOTE：**<br>■ This bit will be cleared when the device wakes up.<br>■ When PM_MODE is 0b00 should not set this bit, Although the hardware won't prevent it. |

### Process data interface reference value register (PMU_PDIREFVAL)

Address offset: 0x1C

Reset value: 0x7654 3210

This register can be accessed by word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PDIVAL[31:0] | When process data interface(PDI) is ready, reading this register returns the reset value, otherwise, other invalid values are returned (not reset value). Used for PDI interface testing. |

# 50.3. Reset and clock unit (RCU)

## 50.3.1. Reset control unit (RCTL)

### Overview

EtherCAT reset control unit includes the control of two kinds of reset: system reset and module reset. System reset includes power-on reset (POR), external pin reset (RSTN) and EtherCAT system reset, which can reset all circuits in the device. Module reset includes digital reset, PHY reset and EtherCAT core reset, which can reset each corresponding module.

### Characteristics

- System reset, reset all circuits in the device.
- Multi-module reset, reset the digital circuit except PHY.
- Single-module reset, reset EtherCAT core and external PHY.

### Function overview

### System reset

System reset can reset the entire device, including power on reset (POR), external pin reset (RSTN) and EtherCAT system level reset, described as follows:

Power-on reset: A power-on reset occurs when the device has just been powered on or when the power is disconnected and reapplied to the device.

RSTN pin reset: Driving the RSTN input pin to low initiates an external pin reset.

EtherCAT system reset: EtherCAT system reset is initiated by a special sequence of three separate consecutive frames/commands.

### Module reset

A module reset affects one or more modules and can generate a reset for a variety of modules, as described below:

Multi-module reset: Performs a digital reset by setting the DRST bit of the configuration register (RCU_RSTCFG). A digital reset resets all submodules of the device except the Ethernet PHY.

Single-module reset: A single-module reset resets only the specified module. Single-module reset does not latch configuration pins and includes port A PHY reset, port B PHY reset, and EtherCAT controller reset.

The single module reset is described as follows:

Port A PHY reset is performed by setting the PHYARST bit in the reset configuration register

(RCU_RSTCFG) or MR_MAIN_REST bit in the PHY control register (PHY_MII_CTL). After port A PHY is reset, the PHYARST bit and soft reset bit are cleared automatically. The other modules of the device are not affected by this reset. The completion of the PHY reset on port A can be determined by whether the PHYARST bit in the polling reset configuration register (RCU_RSTCFG) or the MR_MAIN_REST bit in the PHY control register (PHY_MII_CTL) is cleared.

Port B PHY reset is performed by setting the PHYBRST bit in the reset configuration register (RCU_RSTCFG) or MR_MAIN_REST bit in the PHY control register (PHY_MII_CTL). After the PHY of port B is reset, the PHYBRST bit and soft reset bit are cleared automatically. The other modules of the device are not affected by this reset. The completion of the PHY reset on port B can be determined by whether the PHYBRST bit in the polling reset configuration register (RCU_RSTCFG) or the MR_MAIN_REST bit in the PHY control register (PHY_MII_CTL) is cleared.

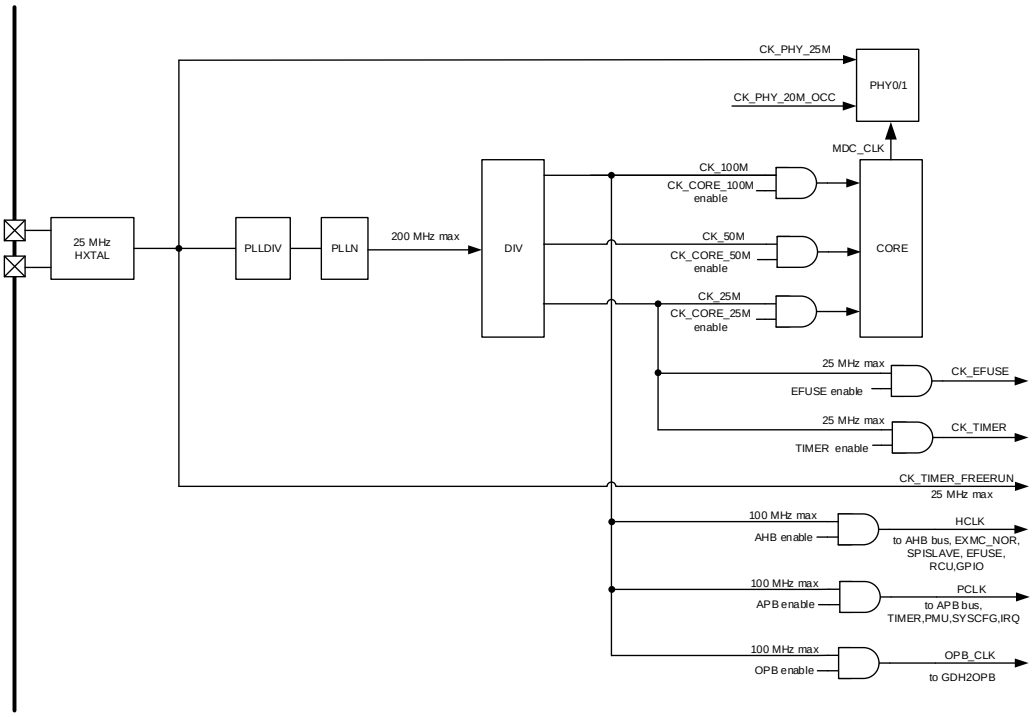
An individual reset of the EtherCAT controller can be performed by resetting ESCRST bit in the configuration register (RCU_RSTCFG). This will reset the EtherCAT core and its registers.

### 50.3.2. Clock control unit (CCTL)

#### Overview

The EtherCAT clock control unit consists primarily of an external High Speed crystal oscillator (HXTAL) and a phase-locked loop (PLL). This clock is usually provided by the OSCIN and OSCOUT of the passive 25MHz crystal oscillator or by the OSCIN pin of the single-ended 25MHz clock source driver.

**Figure 50-3. Clock tree**




## Characteristics

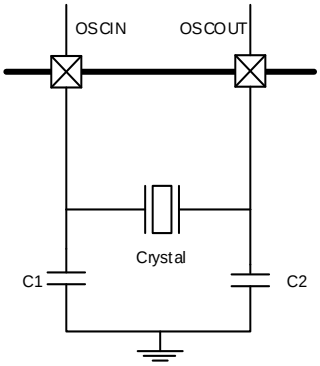
- 25 MHz High speed crystal oscillator (HXTAL).
- A phase locked loop (PLL).

## Function overview

## High speed crystal oscillator (HXTAL)

The device requires a fixed frequency 25 MHz clock source for use by the internal clock oscillator and PLL. This is usually provided by connecting the 25 MHz crystal oscillator to the OSCIN and OSCOUT pins of the chip. This clock can also be provided by using a single-ended 25 MHz clock source driven OSCIN input pin. If a single-ended source is selected, the clock input must run continuously for the device to function properly. Power-saving mode allows the oscillator or external clock input to pause.

**Figure 50-4. HXTAL clock source**




The HXTALSTB flag in clock configuration register (RCU_CLKCFG) indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be released for use until this HXTALSTB bit is set by the hardware. This specific delay period is known as the oscillator “Start-up time”. At this point the HXTAL clock can be used directly as the PLL input clock.

### Phase locked loop (PLL)

The PLL input is a 25MHz HXTAL clock, and after PLLN (8) frequency doubling, CK_PLL (200MHz) is obtained. The CK_PLL clock is divided 2 / 4/ 8 by PLLDIV to obtain 100MHz, 50MHz, 25MHz clocks for EtherCAT kernel, and corresponding clocks can be turned on in the core enable register (RCU_COREEN). The clock of the corresponding module can be enabled in the AHB enable register (RCU_AHBEN), APB enable register (RCU_APBEN) and core enable register (RCU_COREEN).

## 50.3.3. Register definition

RCU base address: 0x3400

### AHB enable register (RCU_AHBEN)

Address offset: 0x00

Reset value: 0x0000 000F

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | | | | | | | | | GPIOEN | OPBEN |
| | | | | | | | | | | | | | | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | GPIOEN | GPIO clock enable |

This bit is set and reset by software.
0: Disabled GPIO clock
1: Enabled GPIO clock

| | | |
|---|---|---|
| 0 | OPBEN | OPB clock enable<br>This bit is set and reset by software.<br>0: Disabled OPB clock<br>1: Enabled OPB clock |

### APB enable register (RCU_APBEN)

Address offset: 0x04
Reset value: 0x0000 0077

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | TIMERFUNEN | TIMEREN | PMUEN | Reserved | SYSCFGEN | IRQEN | Reserved |
| | | | | | | | | | rw | rw | rw | | rw | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:7 | Reserved | Must be kept at reset value. |
| 6 | TIMERFUNEN | TIMER function clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER function clock<br>1: Enabled TIMER function clock |
| 5 | TIMEREN | TIMER clock enable<br>This bit is set and reset by software.<br>0: Disabled TIMER clock<br>1: Enabled TIMER clock (CK_TIMER) |
| 4 | PMUEN | PMU clock enable<br>This bit is set and reset by software.<br>0: Disabled PMU clock<br>1: Enabled PMU clock |
| 3 | Reserved | Must be kept at reset value. |
| 2 | SYSCFGEN | SYSCFG clock enable<br>This bit is set and reset by software.<br>0: Disabled SYSCFG clock<br>1: Enabled SYSCFG clock |

| | | |
|---|---|---|
| 1 | IRQEN | IRQ clock enable<br>This bit is set and reset by software.<br>0: Disabled IRQ clock<br>1: Enabled IRQ clock |
| 0 | Reserved | Must be kept at reset value. |

### Core enable register (RCU_COREEN)

Address offset: 0x08
Reset value: 0x0000 0007

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | CORE100MEN | CORE50MEN | CORE25MEN |
| | | | | | | | | | | | | | rw | rw | rw |

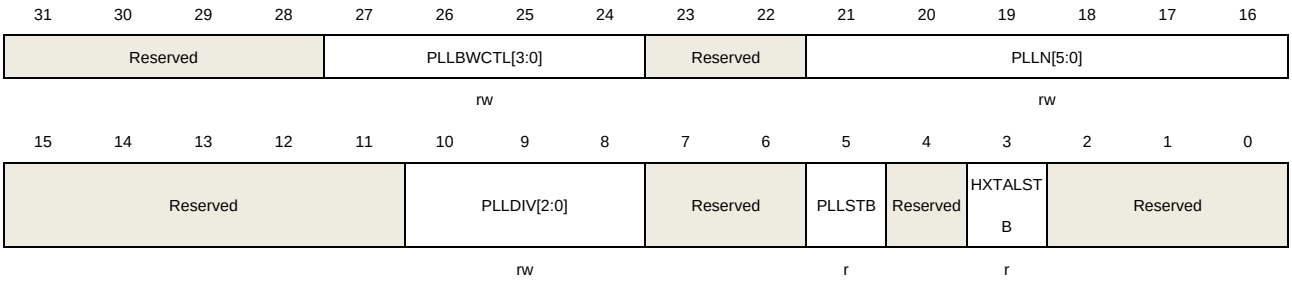
| Bits | Fields | Descriptions |
|---|---|---|
| 31:3 | Reserved | Must be kept at reset value. |
| 2 | CORE100MEN | EtherCAT core 100M clock enable<br>This bit is set and reset by software.<br>0: Disabled CK_CORE_100M clock<br>1: Enabled CK_CORE_100M clock |
| 1 | CORE50MEN | EtherCAT core 50M clock enable<br>This bit is set and reset by software.<br>0: Disabled CK_CORE_50M clock<br>1: Enabled CK_CORE_50M clock |
| 0 | CORE25MEN | EtherCAT core 25M clock enable<br>This bit is set and reset by software.<br>0: Disabled CK_CORE_25M clock<br>1: Enabled CK_CORE_25M clock |

### Clock configuration register (RCU_CLKCFG)

Address offset: 0x0C
Reset value: 0x0528 0400

**Note:** PLLBWCTL, PLLN and PLLDIV can only be read and written when PLL_CFG_KEY = 1.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:28 | Reserved | Must be kept at reset value. |
| 27:24 | PLLBWCTL[3:0] | PLL band width control signal |
| 23:22 | Reserved | Must be kept at reset value. |
| 21:16 | PLLN[5:0] | PLL clock multiplication factor<br>This bit is set and reset by software.<br>000000: Reserved<br>000001: Reserved<br>…<br>000110: Reserved<br>000111: Reserved<br>001000: Multiplication factor is 8<br>001001: Multiplication factor is 9<br>001010: Multiplication factor is 10<br>…<br>111110: Multiplication factor is 62<br>111111: Multiplication factor is 63 |
| 15:11 | Reserved | Must be kept at reset value. |
| 10:8 | PLLDIV[2:0] | PLL clock frequency division factor<br>This bit is set and reset by software.<br>000: Division factor is 1<br>001: Division factor is 2<br>…<br>110: Division factor is 7<br>111: Division factor is 8 |
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | PLLSTB | PLL clock stabilization flag<br>Set by hardware to indicate if the PLL output clock is stable and ready for use.<br>0: PLL is not stable<br>1: PLL is stable |
| 4 | Reserved | Must be kept at reset value. |

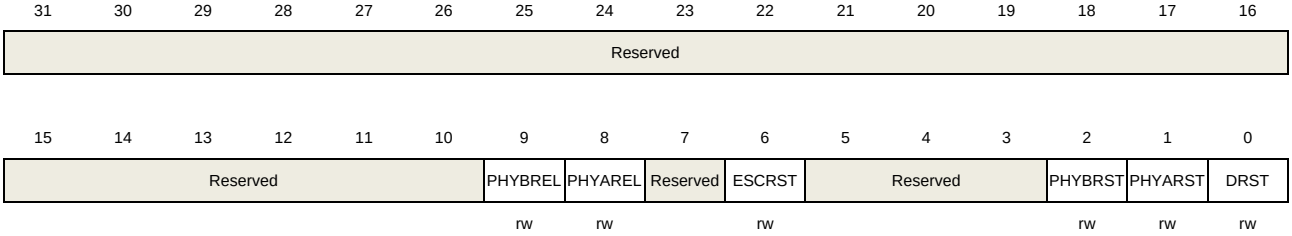
| | | |
|---|---|---|
| 3 | HXTALSTB | High speed crystal oscillator (HXTAL) clock stabilization flag<br>Set by hardware to indicate if the HXTAL oscillator is stable and ready for use.<br>0: HXTAL oscillator is not stable<br>1: HXTAL oscillator is stable |
| 2:0 | Reserved | Must be kept at reset value. |

### Reset configuration register (RCU_RSTCFG)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:10 | Reserved | Must be kept at reset value. |
| 9 | PHYBREL | Port B PHY release<br>0: Port B PHY remains in the reset state<br>1: Port B PHY releases from reset<br>**Note**: This bit is valid when phyrst_mode = 1 |
| 8 | PHYAREL | Port A PHY release<br>0: Port A PHY remains in the reset state<br>1: Port A PHY releases from reset<br>**Note**: This bit is valid when phyrst_mode = 1 |
| 7 | Reserved | Must be kept at reset value. |
| 6 | ESCRST | EtherCAT reset<br>Setting this bit to 1 will reset the EtherCAT core. When the EtherCAT core is released from the reset state, this bit is automatically cleared by the hardware. When this bit is set, all writes to this bit are ignored. |
| 5:3 | Reserved | Must be kept at reset value. |
| 2 | PHYBRST | Port B PHY reset<br>Setting this bit to 1 will reset the port B PHY. When port B PHY is released from the reset state, this bit is automatically cleared by the hardware. When the bit is set, all writes to that bit are ignored. |
| 1 | PHYARST | Port A PHY reset<br>Setting this bit to 1 will reset the port A PHY. When port A PHY is released from the |

reset state, this bit is automatically cleared by the hardware. When the bit is set, all writes to that bit are ignored.

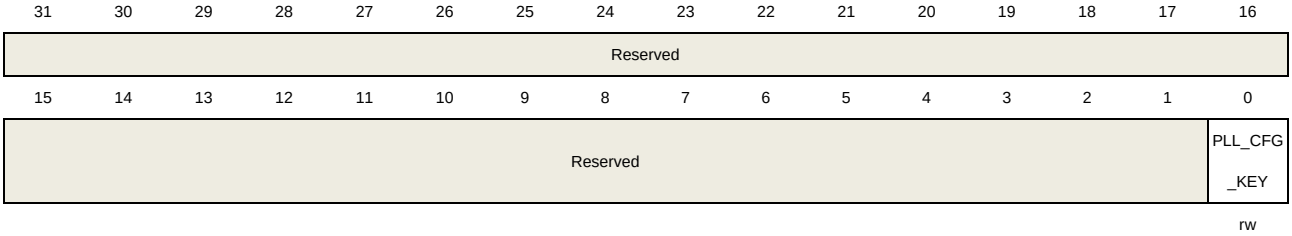
| | | |
|---|---|---|
| 0 | DRST | Digital reset<br>Setting this bit to 1 will reset the entire chip (except for the PLL, port B PHY and port A PHY). When the chip is released from the reset state, this bit is automatically cleared by the hardware. When this bit is set, all writes to this bit are ignored. |

### PLL configuration key register (RCU_PLL_CFG_KEY)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).



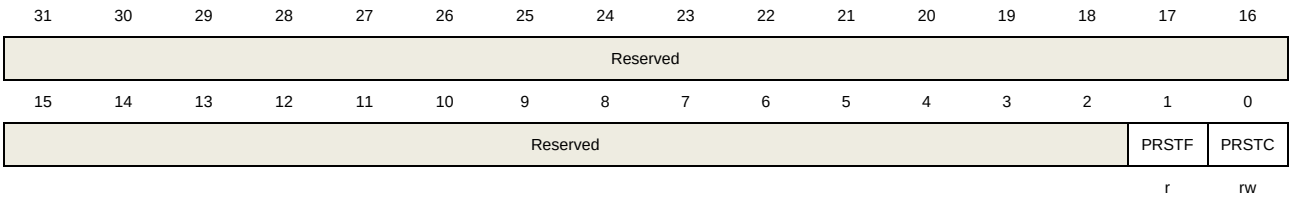

| Bits | Fields | Descriptions |
|---|---|---|
| 31:1 | Reserved | Must be kept at reset value. |
| 0 | PLL_CFG_KEY | When the register is written to 0x78b465a1, the bit is 1 and PLLBWCTL, PLLN and PLLDIV bits in RCU_CLKCFG register can be read and written. |

### Pin reset flag register (RCU_PRSTF)

Address offset: 0x18

Reset value: 0x0000 0002

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:2 | Reserved | Must be kept at reset value. |
| 1 | PRSTF | Pin reset flag<br>This bit is set when pin reset occurs and cleared when PRSTC is set.<br>0: No pin reset occurs<br>1: Pin reset occurs |

| | | |
|---|---|---|
| 0 | PRSTC | Pin reset flag clear<br>Write 1 by software to reset the PRSTF flag |

# 50.4. Interrupt controller (INTC)

## 50.4.1. Overview

The multi-layer interrupt structure of the device is programmable and controlled by the interrupt controller (INTC). Interrupt events are generated internally by individual submodules and can be configured to output a single external host interrupt via the IRQ pin.

## 50.4.2. Characteristics

- The IRQ interrupt buffer mode, polarity, and de-assertion interval can be modified.
- The IRQ interrupt can be set the output mode to open-drain, enabling multiple devices to share the interrupt.
- All internal interrupts can be masked and trigger the IRQ interrupt.
- The device supports the following 8 types interrupts:
  - Software interrupt.
  - Device ready interrupt.
  - Ethernet PHY interrupt.
  - Timer interrupt.
  - PME interrupt.
  - AHB2OPB bridge interrupt.
  - EtherCAT interrupt.
  - Clock output test mode.

## 50.4.3. Interrupts function overview

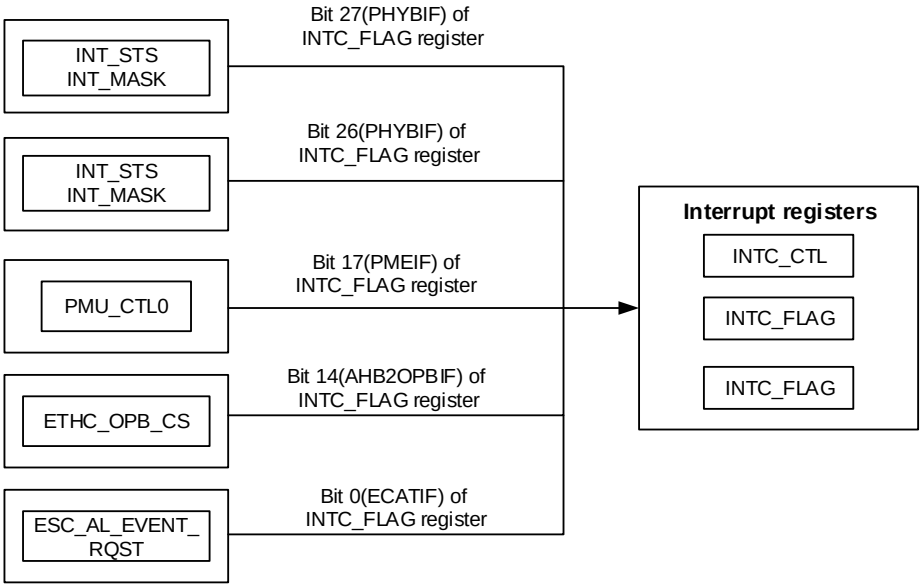
The interrupt of the device can be divided into the following two types according to whether the interrupt source is enabled and cleared in the register of the submodule:

- The first type includes software, device ready, and timer interrupts that are directly accessed and configured (including monitored, enabled/disabled, and cleared) through INTC_FLAG register and INTC_EN register.
- The second type includes Ethernet PHY, power management, AHB2OPB bridge, and EtherCAT interrupts. INTC_FLAG register can provide indications of these interrupt events, but has no specific information of interrupt source. Software needs to poll an submodule interrupt register of the to determine the interrupt source. INTC_FLAG register can be cleared only after the interrupt has been processed and the interrupt source cleared.

Interrupt events can trigger external IRQ interrupt pin output. By configuring INTC_CTL register, user can enable/disable IRQ interrupt pin output and configure IRQ interrupt buffer mode, polarity and de-assertion interval. DEAS field of INTC_CTL register is used to configure the interrupt request de-assertion interval, which guarantees that the minimum IRQ interrupt output de-assertion interval period, and that de-assertion interval always starts when IRQ pin is set to de-assertion. The relationship between interrupt register and interrupt source control

register, as shown in ***Figure 50-5. Block diagram of interrupt***.

**Figure 50-5. Block diagram of interrupt**




### Software interrupt

Interrupt controller provides control over a general purpose software interrupt. When SWIE bit of INTC_EN register is switched from 0 to 1, SWIF bit of INTC_FLAG register is set. This interrupt provides a relatively simple method of generating interrupts in software and is used in conventional software design.

In order for a software interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### Device ready interrupt

Interrupt controller provides control over a device ready interrupt. When READYIE bit of INTC_EN register is switched from 0 to 1, READYIF bit of INTC_FLAG register is used to indicate that the device is ready for access after power-on or reset condition.

In order for a device ready interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### Ethernet PHY interrupt

Interrupt controller provides control over a ethernet PHY interrupt. When PHYAIE bit of INTC_EN register is switched from 0 to 1, PHYAIF and PHYBIF bits of INTC_FLAG register are used to indicate interrupt events from the Ethernet PHY. For more information about Ethernet PHY interrupt sources, refer to the ***Ethernet PHYS***.

In order for a ethernet PHY interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### Timer interrupt

Interrupt controller provides control over a timer interrupt. This interrupt is generated when the value of timer count register changes from 0 to 0xFFFF. When TIMIE bit of INTC_EN register is switched from 0 to 1, TIMIF bit of INTC_FLAG register is used to indicate interrupt events from the timer.

In order for a timer interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### PME interrupt

Interrupt controller provides control over a PME interrupt. When PMEIE bit of INTC_EN register is switched from 0 to 1, PMEIF bit of INTC_FLAG register is used to indicate interrupt events from the PMU. For more information about PMU interrupt sources, refer to the ***Power management unit (PMU)***.

In order for a ethernet PHY interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### AHB2OPB bridge interrupt

Interrupt controller provides control over a AHB2OPB bridge interrupt. When AHB2OPBIE bit of INTC_EN register is switched from 0 to 1, AHB2OPBIF bit of INTC_FLAG register is used to indicate the AHB2OPB bridge interrupt event from the BUS. For more information about SYS interrupt sources, refer to the ***System and bus architecture***.

In order for a AHB2OPB bridge interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### EtherCAT interrupt

Interrupt controller provides control over a EtherCAT interrupt. When ECATIE bit of INTC_EN register is switched from 0 to 1, ECATIF bit of INTC_FLAG register is used to indicate interrupt events from the EtherCAT. For more information about EtherCAT interrupt sources, refer to the ***EtherCAT***.

In order for a EtherCAT interrupt event to trigger the external IRQ interrupt pin, IRQ output must be enabled by setting IRQEN bit of INTC_CTL register to 1.

### Clock output test mode

In order to debug system and observe the clock, the IRQ pin output crystal oscillator clock can be realized by setting IRQCKOUT bit of INTC_CTL register to 1. At this point, the IRQ pin must be configured in push-pull output mode (IRQMODE=1) for best results.

## 50.4.4. Register definition

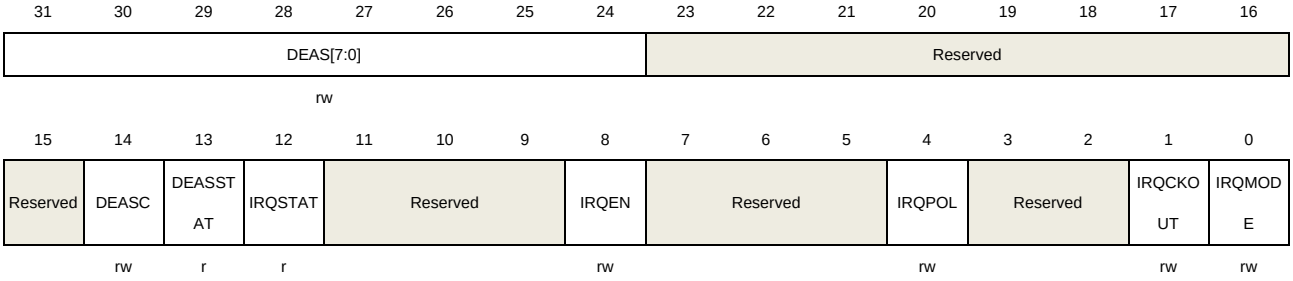
INTC base address: 0x0000 3A00

### Control register (INTC_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | DEAS[7:0] | Interrupt de-assertion interval<br>These bits are used to configure the de-assertion interval, uints in 10us.<br>When setting these bits to 0, which disables the DEAS interval, reset the interval counter, and send any pending interrupts. If these bits are set to a non-zero value, any subsequent interrupts will follow this setting. |
| 23:15 | Reserved | Must be kept at reset value. |
| 14 | DEASC | Interrupt de-assertion interval clear<br>When setting this bit is to 1, the de-assertion interval counter will be cleared and a new de-assertion interval count will be enabled (regardless of whether the de-assertion interval is currently active or not). This bit is automatically cleared by hardware. This bit is not reset when a software reset occurs on the device.<br>0: No effect<br>1: Clear the de-assertion interval counter |
| 13 | DEASSTAT | Interrupt de-assertion interval status<br>This bit is used to indicate the status of the interrupt de-assertion interval.<br>0: Not in de-assertion interval (interrupts will be sent to the IRQ pin)<br>1: In de-assertion interval (interrupts will not be sent to the IRQ pin) |
| 12 | IRQSTAT | Internal IRQ line status<br>This bit is used to indicate the status of the internal IRQ line and is not affected by the setting of the IRQEN bit.<br>0: None of the enabled interrupts active<br>1: At least one enabled interrupt active |
| 11:9 | Reserved | Must be kept at reset value. |

| | | |
|---|---|---|
| 8 | IRQEN | IRQ pin output enable<br>This bit controls the interrupt output function of the IRQ pin.<br>0: Disable IRQ pin output<br>1: Enable IRQ pin output |
| 7:5 | Reserved | Must be kept at reset value. |
| 4 | IRQPOL | IRQ pin output polarity<br>This bit is not reset when a digital reset occurs on the device. This bit is reset when a system reset occurs on the device, such as power-on reset, pin reset and EtherCAT system reset.<br>0: IRQ pin output activation level is low<br>1: IRQ pin output activation level is high<br>**Note:** When IRQ pin is set to output open-drain mode (IRQMODE=0), this bit is ignored and the IRQ pin output activation level is always low. |
| 3:2 | Reserved | Must be kept at reset value. |
| 1 | IRQCKOUT | IRQ clock output<br>0: No clock output<br>1: IRQ pin output crystal oscillator clock (Used for system debugging, observing the clock)<br>**Note:** When this bit is set to 1, the IRQ pin must be set to push-pull output mode (IRQMODE=1). |
| 0 | IRQMODE | IRQ pin output mode<br>This bit is not reset when a digital reset occurs on the device.This bit is reset when a system reset occurs on the device, such as power-on reset, pin reset and EtherCAT system reset.<br>0: Output open-drain mode<br>1: Output push-pull mode |

### Flag register (INTC_FLAG)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWIF | READYIF | Reserved | | PHYBIF | PHYAIF | Reserved | | | | | | TIMIF | Reserved | PMEIF | Reserved |
| rc_w1 | rc_w1 | | | r | r | | | | | | | rc_w1 | | rc_w1 | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | AHB2OPBIF | Reserved | | | | | | | | | | | | | ECATIF |
| | r | | | | | | | | | | | | | | r |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31 | SWIF | Software interrupt flag<br>This interrupt is generated when SWIE bit in INTC_EN register is 1.<br>The software can clear it by writing 1. |
| 30 | READYIF | Device ready interrupt flag<br>This interrupt is used to indicate that the device is ready for access after power-on or reset condition.<br>The software can clear it by writing 1. |
| 29:28 | Reserved | Must be kept at reset value. |
| 27 | PHYBIF | Ethernet PHY B interrupt flag<br>This bit indicates an Ethernet PHY B interrupt event. |
| 26 | PHYAIF | Ethernet PHY A interrupt flag<br>This bit indicates an Ethernet PHY A interrupt event. |
| 25:20 | Reserved | Must be kept at reset value. |
| 19 | TIMIF | Timer interrupt flag<br>This interrupt is generated when the timer counter register changes from 0 to 0xFFFF.<br>The software can clear it by writing 1. |
| 18 | Reserved | Must be kept at reset value. |
| 17 | PMEIF | PME interrupt flag<br>This interrupt is generated when the PMU detects a power management event configured in the PMU_CTL register.<br>The software can clear it by writing 1.<br>**Note:** Interrupt de-assertion interval does not apply to PMU interrupt. |
| 16:15 | Reserved | Must be kept at reset value. |
| 14 | AHB2OPBIF | AHB2OPB bridge interrupt flag<br>This bit indicates an AHB2OPB bridge interrupt event from the BUS. |
| 13:1 | Reserved | Must be kept at reset value. |
| 0 | ECATIF | EtherCAT interrupt flag<br>This bit indicates an interrupt event from EtherCAT. |

### Enable register (INTC_EN)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SWIE | READYIE | Reserved | | PHYBIE | PHYAIE | Reserved | | | | | | TIMIE | Reserved | PMEIE | Reserved |
| rw | rw | | | rw | rw | | | | | | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | AHB2OPBIE | Reserved | | | | | | | | | | | | | ECATIE |
| | rw | | | | | | | | | | | | | | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | SWIE | Software interrupt enable<br>0: Disable software interrupt<br>1: Enable software interrupt |
| 30 | READYIE | Device ready interrupt enable<br>0: Disable device ready interrupt<br>1: Enable device ready interrupt |
| 29:28 | Reserved | Must be kept at reset value. |
| 27 | PHYBIE | Ethernet PHY B interrupt enable<br>0: Disable ethernet PHY B interrupt<br>1: Enable ethernet PHY B interrupt |
| 26 | PHYAIE | Ethernet PHY A interrupt enable<br>0: Disable ethernet PHY A interrupt<br>1: Enable ethernet PHY A interrupt |
| 25:20 | Reserved | Must be kept at reset value. |
| 19 | TIMIE | Timer interrupt enable<br>0: Disable timer interrupt<br>1: Enable timer interrupt |
| 18 | Reserved | Must be kept at reset value. |
| 17 | PMEIE | PME interrupt enable<br>0: Disable power management interrupt<br>1: Enable power management interrupt |
| 16:15 | Reserved | Must be kept at reset value. |
| 14 | AHB2OPBIE | AHB2OPB bridge interrupt enable<br>0: Disable AHB2OPB bridge interrupt<br>1: Enable AHB2OPB bridge interrupt |
| 13:1 | Reserved | Must be kept at reset value. |
| 0 | ECATIE | EtherCAT interrupt enable<br>0: Disable EtherCAT interrupt<br>1: Enable EtherCAT interrupt |

# 50.5. General-purpose I/Os (GPIO)

## 50.5.1. Overview

There are up to 35 general purpose I/O pins (GPIO), Each GPIO port will determine the current capabilities of that port based on the current operating mode of the chip, including input/output modes. Each of the GPIO pins can be configured as a pull-up/pull-down or floating. When the pin is in output mode, the pin can be configured as a push-pull/drain open/source open drain output.

## 50.5.2. Characteristics

- Each pin weak pull-up/pull-down function.
- Output push-pull/open-drain enable control.
- Configures the function of the selected pin according to the chip mode.

## 50.5.3. Function overview

### GPIO pin configuration

When is the reset, All the GPIO ports are configured as the input floating mode that input disabled without pull-up(PU)/pull-down(PD) resistors. After the chip is reset, wait for the EEPROM to load. When the loading is complete, determine the initial state of the pin after the chip is reset according to ESC PDI_TYPE. When PDI_TYPE equal to 0x80 select SPI mode.

The GPIO pin is controlled as input or output state according to the working state.
All GPIO pins have an internal weak pull-up and weak pull-down option When the GPIO pin is configured as an output pin, it can configure the output drive mode: push-pull or drain open drain and source open drain mode. The pull-down mode and output mode configurations support writing via SPI communication.

### External interrupt/event lines

Only one external interrupt output interface is supported. The interrupt output configuration is determined by the internal register of the chapter ***Interrupt controller (INTC)***, and the output mode is also determined by the internal bit.

### Alternate functions (AF)

When the chip is in different modes, each pin has different functions.

SPI mode: When PDI_TYPE equal to 0x80 and the pad of MCU_PDI_TYPE equal to 0. AFIO is adjusted to SPI mode.

**Table 50-4. GPIO configuration table**

| Mode name | Register/signal | Description |
|---|---|---|

| | | |
|---|---|---|
| spi8w_gpio | pdi_type / mcu_pdi_type / spi_ext_mode / sip_mode / line_mode | pdi_type == 0x80;<br>mcu_pdi_type == 1'b0;<br>sip_mode == 1'b0;<br>line_mode == 2'b11;<br>LINKACTLED0 Pin must be externally drop-down<br>LINKACTLED1 Pin must be externally drop-down |
| spi8w_mii_down | pdi_type / mcu_pdi_type / spi_ext_mode / sip_mode / line_mode / chip_mode | pdi_type == 0x80;<br>mcu_pdi_type == 1'b0;<br>sip_mode == 1'b0;<br>line_mode == 2'b10;<br>chip_mode == 2'b10<br>LINKACTLED0 Pin must be externally drop-down<br>LINKACTLED1 Pin must be externally pull-up |
| spi8w_mii_up | pdi_type / mcu_pdi_type / spi_ext_mode / sip_mode / line_mode / chip_mode | pdi_type == 0x80;<br>mcu_pdi_type == 1'b0;<br>sip_mode == 1'b0;<br>line_mode == 2'b10;<br>chip_mode == 2'b11;<br>LINKACTLED0 Pin must be externally pull-up<br>LINKACTLED1 Pin must be externally pull-up |

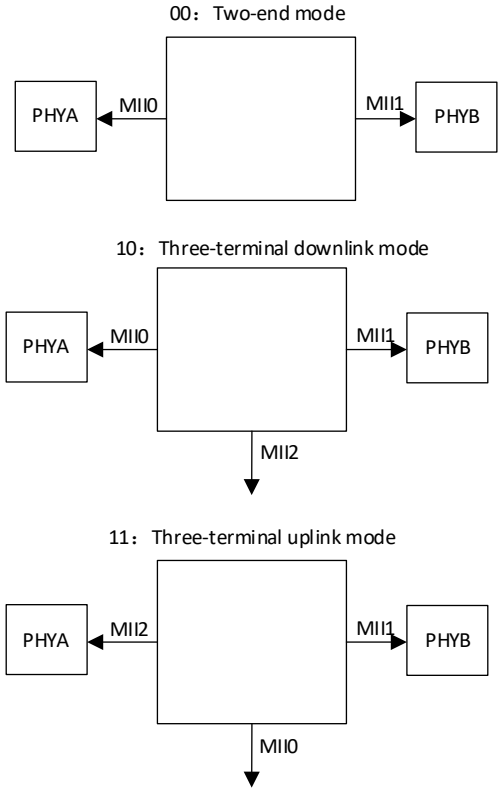
In addition, when in SPI (2/4/8 wire) +MII and the chip_mode [1:0] is not equal to 0x11, the EtherCAT port 0 is connected to the internal PHY A.

When chip_mode [1:0] is equal to 0x11, the EtherCAT port 0 is connected to the MII pin, port 2 is connected to the internal PHY A.

When chip_mode [1:0] is equal to 0x10b, the EtherCAT port 2 is connected to the MII pin.

When chip_mode [1:0] is equal to 0x00, In this case, the output of SPI+GPIO mode is not affected, and the MII signal is not output in SPI+MII mode.

**Figure 50-6. Port line PHYS**




**Note:**

1. Some pins are locked during power-on reset or when RST# is set to invalid, and automatically switch after being locked.
2. The MII_LINKPOL signal is latched after reset to determine the polarity of the MII_LINK pin. If MII_LINKPOL equals 0, it indicates that the MII_LINK pin is active low, signifying that a 100 Mbps full-duplex link has been established. If MII_LINKPOL equals 1, it indicates that the MII_LINK pin is active high, signifying that a 100 Mbps full-duplex link has been established.
3. SYNC1_LATCH1/SYNC0_LATCH0 pad omode / io_en is determined by the ESC internal register.
4. The following latch signals must to be pull up or down in the following mode, and cannot be set to the X state.

   (1). If spi_ext_mode is equal to 1 and inphy_bypass is equal to 0, the chip_mode[1:0] must be set drop-down state.

   (2). The pad of IO16 must be configure as the pull-up or drop-down state

   (3). The pad of EESIZE must be configure as the pull-up or drop-down state

   (4). The pad of IO17 must be configure as the drop-down and drop-down state

   (5). When inphy_bypass is equal to 1, the pad of MII_LINKPOL must be set the pull-up

or drop-down state.

PDI_TYPE: Reference to EtherCAT register ***ESC PDI Control register (ESC_PDI_CTL)***.

line_mode: The SPI output to GPIO is determined by the SPI input instruction

spi_ext_mode: Reference to System configuration register in ***System configuration register 0 (ESC_SYSCFG_CFG0)***.

chip_mode[1:0]: pad of LINKACTLED1/ LINKACTLED0 latched after reset.

inphy_bypass: The register bit configured by factory set to 0,cannot be modified.

### Analog configuration

When GPIO pin is used as analog configuration:

- The weak pull-up and pull-down resistors are disabled.
- The output buffer is disabled.
- The schmitt trigger input is de-activated.
- Read access to the port input status register gets the value “0”.

***Figure 50-7. Basic structure of Analog configuration*** shows the analog configuration of the GPIO pin.

**Figure 50-7. Basic structure of Analog configuration**




### Alternate function (AF) configuration

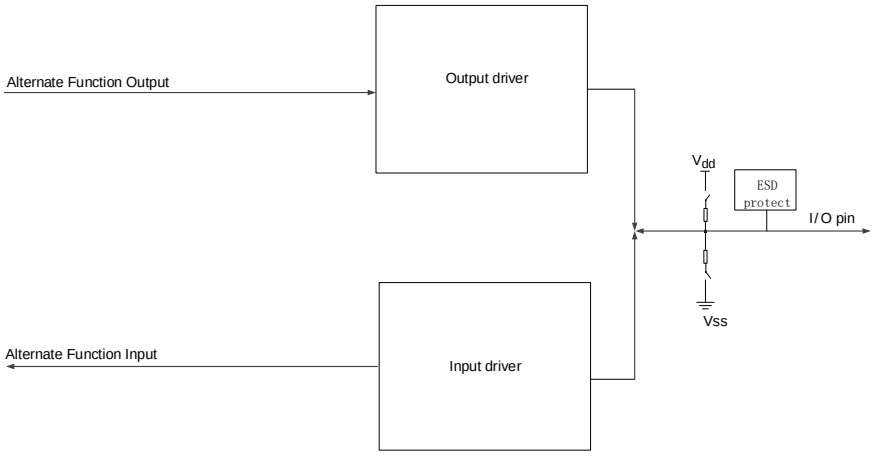
To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

When be configured as alternate function:

- The output buffer is enabled in open-drain or push-pull configuration.
- The output buffer is driven by the peripheral.
- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.

***Figure 50-8. Basic structure of Alternate function configuration*** shows the alternate function configuration of the GPIO pin.

**Figure 50-8. Basic structure of Alternate function configuration**




**Note:**

In OSPI mode, pdi_gpio15 cannot be used in OSPI+GPIO mode due to the large number of SPI pins occupied. ESC_CLKOUT_25M cannot be used in OSPI+MII mode.

## 50.5.4. Register definition

GPIO base address: 0x3500

### Port output mode register0 (GPIO0_OMODE0)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 30 | 29 28 | 27 26 | 25 24 | 23 22 | 21 20 | 19 18 | 17 16 |
|---|---|---|---|---|---|---|---|
| OM015[1:0] | OM014[1:0] | OM013[1:0] | OM012[1:0] | OM011[1:0] | OM010[1:0] | Reserved | OM008[1:0] |
| rw | rw | rw | rw | rw | rw | | rw |

| 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|
| OM007[1:0] | OM006[1:0] | OM005[1:0] | OM004[1:0] | Reserved | OM002[1:0] | OM001[1:0] | OM000[1:0] |
| rw | rw | rw | rw | | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | OM015[1:0] | Pin IO11 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 29:28 | OM014[1:0] | Pin OE_EXT output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 27:26 | OM013[1:0] | Pin IO4 output mode bit<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to OM000[1:0] description |
| 25:24 | OM012[1:0] | Pin IO5 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 23:22 | OM011[1:0] | Pin IO6 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 21:20 | OM010[1:0] | Pin LATCH_IN output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 19:18 | Reserved | Must be kept at reset value. |
| 17:16 | OM008[1:0] | Pin WD_STATE output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 15:14 | OM007[1:0] | Pin IO7 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 13:12 | OM006[1:0] | Pin IO8 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 11:10 | OM005[1:0] | Pin EOF output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 9:8 | OM004[1:0] | Pin SOF output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 7:6 | Reserved | Must be kept at reset value |
| 5:4 | OM002[1:0] | Pin IO18 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 3:2 | OM001[1:0] | Pin IO17 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 1:0 | OM000[1:0] | Pin IO16 output mode bit<br>These bits are set and cleared by software.<br>00: Output push-pull mode (reset value)<br>01: Output open-drain mode<br>10: Output open-source mod |

11: reserved

### Port output mode register1 (GPIO0_OMODE1)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | OM017[1:0] | | OM016[1:0] | |
| | | | | | | | | | | | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:2 | OM017[1:0] | Pin IO13 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 1:0 | OM016[1:0] | Pin IO12 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |

### Port output mode register2 (GPIO1_OMOD0)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OM115 | | OM114 | | OM113 | | Reserved | | OM111 | | OM110 | | Reserved | | OM108 | |
| rw | | rw | | rw | | | | rw | | rw | | | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OM107 | | OM106 | | OM105 | | Reserved | | OM103 | | OM102 | | OM101 | | OM100 | |
| rw | | rw | | rw | | | | rw | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | OM115[1:0] | Pin LINKACTLED0 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 29:28 | OM114[1:0] | Pin LINKACTLED1 output mode bit<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to OM000[1:0] description |
| 27:26 | OM113[1:0] | Pin EESIZE output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 25:24 | Reserved | Must be kept at reset value |
| 23:22 | OM111[1:0] | Pin EESCL output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 21:20 | OM110[1:0] | Pin EESDA output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 19:18 | Reserved | Must be kept at reset value |
| 17:16 | OM108[1:0] | Pin IO2 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 15:14 | OM107[1:0] | Pin IO1 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 13:12 | OM106[1:0] | Pin IO0 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 11:10 | OM105[1:0] | Pin WD_TRIG output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 9:8 | Reserved | Must be kept at reset value |
| 7:6 | OM103[1:0] | Pin IO9 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 5:4 | OM102[1:0] | Pin IO15 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 3:2 | OM101[1:0] | Pin IO14 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 1:0 | OM100[1:0] | Pin IO10 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |

### Port output mode register3 (GPIO1_OMOD1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | OM117 | | OM116 | |
| | | | | | | | | | | | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:2 | OM117[1:0] | Pin OUTVALID output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |
| 1:0 | OM116[1:0] | Pin IO3 output mode bit<br>These bits are set and cleared by software.<br>Refer to OM000[1:0] description |

### Port pull-up/down register0 (GPIO0_PUD0)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PUD015[1:0] | | PUD014[1:0] | | PUD013[1:0] | | PUD012[1:0] | | PUD011[1:0] | | PUD010[1:0] | | PUD009[1:0] | | PUD008[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PUD007[1:0] | | PUD006[1:0] | | PUD005[1:0] | | PUD004[1:0] | | Reserved | | PUD002[1:0] | | PUD001[1:0] | | PUD000[1:0] | |
| rw | | rw | | rw | | rw | | | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | PUD015[1:0] | Pin IO11 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 29:28 | PUD014[1:0] | Pin OE_EXT pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 27:26 | PUD013[1:0] | Pin IO4 pull-up or pull-down bits |

| | | |
|---|---|---|
| | | These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 25:24 | PUD012[1:0] | Pin IO5 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 23:22 | PUD011[1:0] | Pin IO6 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 21:20 | PUD010[1:0] | Pin LATCH_IN pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 19:18 | PUD009[1:0] | Pin SYNC1_LATCH1 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 17:16 | PUD008[1:0] | Pin WD_STATE pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 15:14 | PUD007[1:0] | Pin IO7 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 13:12 | PUD006[1:0] | Pin IO8 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 11:10 | PUD005[1:0] | Pin EOF pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 9:8 | PUD004[1:0] | Pin SOF pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 7:6 | Reserved | Must be kept at reset value |
| 5:4 | PUD002[1:0] | Pin IO18 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 3:2 | PUD001[1:0] | Pin IO17 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 1:0 | PUD000[1:0] | Pin IO16 pull-up or pull-down bits<br>These bits are set and cleared by software. |

00: Floating mode, no pull-up and pull-down (reset value)

01: With pull-up mode

10: With pull-down mode

11: analog mode

### Port pull-up/down register1 (GPIO0_PUD1)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | PUD017[1:0] | | PUD016[1:0] | |
| | | | | | | | | | | | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:2 | PUD017[1:0] | Pin IO13 pull-up or pull-down bits<br>These bits are set and cleared by so ftware.<br>Refer to PUD000[1:0] description |
| 1:0 | PUD016[1:0] | Pin IO12 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |

### Port pull-up/down register2 (GPIO1_PUD0)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PUD115[1:0] | | PUD114[1:0] | | PUD113[1:0] | | PUD112[1:0] | | PUD111[1:0] | | PUD110[1:0] | | PUD109[1:0] | | PUD108[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PUD107[1:0] | | PUD106[1:0] | | PUD105[1:0] | | PUD104[1:0] | | PUD103[1:0] | | PUD102[1:0] | | PUD101[1:0] | | PUD100[1:0] | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | PUD115[1:0] | Pin LINKACTLED0 pull-up or pull-down bits<br>These bits are set and cleared by software. |

| | | |
|---|---|---|
| | | Refer to PUD000[1:0] description |
| 29:28 | PUD114[1:0] | Pin LINKACTLED1 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 27:26 | PUD113[1:0] | Pin EESIZE pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 25:24 | PUD112[1:0] | Pin IRQ pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 23:22 | PUD111[1:0] | Pin EESCL pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 21:20 | PUD110[1:0] | Pin EESDA pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 19:18 | PUD109[1:0] | Pin TESTMODE pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 17:16 | PUD108[1:0] | Pin IO2 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 15:14 | PUD107[1:0] | Pin IO1 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 13:12 | PUD106[1:0] | Pin IO0 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 11:10 | PUD105[1:0] | Pin WD_TRIG pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 9:8 | PUD104[1:0] | Pin SYNC0_LATCH0 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 7:6 | PUD103[1:0] | Pin IO9 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |

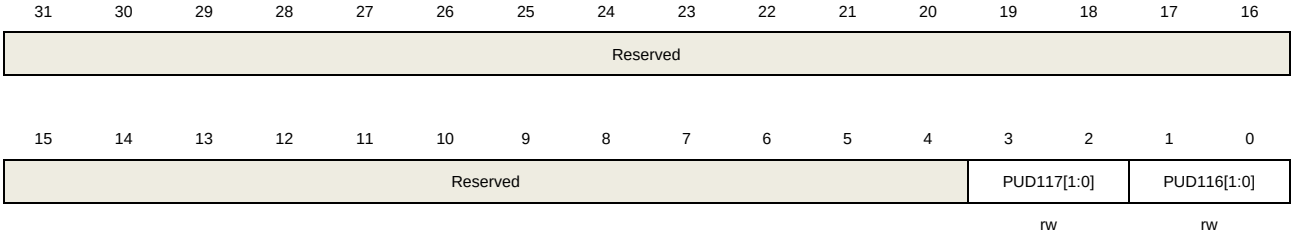
| | | |
|---|---|---|
| 5:4 | PUD102[1:0] | Pin IO15 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD100[1:0] description |
| 3:2 | PUD101[1:0] | Pin IO14 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 1:0 | PUD100[1:0] | Pin IO10 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |

#### Port pull-up/down register3 (GPIO1_PUD1)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:4 | Reserved | Must be kept at reset value |
| 3:2 | PUD117[1:0] | Pin OUTVALID pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |
| 1:0 | PUD116[1:0] | Pin IO3 pull-up or pull-down bits<br>These bits are set and cleared by software.<br>Refer to PUD000[1:0] description |

# 50.6. TIMER

## 50.6.1. Basic Timer

### Overview

The basic timer module has a 16-bit counter that can be used as an unsigned counter. The basic timer can be configured to generate interrupts. The resolution of basic timer is 100 µs.

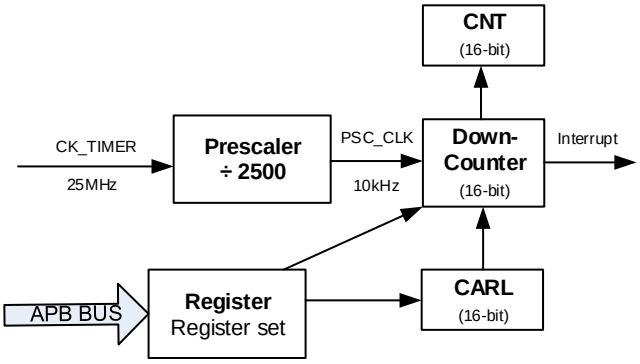
## Characteristics

- Counter width: 16 bits.
- Source of count clock is internal clock only.
- Counter mode: count down.
- Resolution: 100 µs.
- Auto-reload function.
- Interrupt output: update event.

## Block diagram

*Figure 24-126. Basic timer block diagram* provides details on the internal configuration of the basic timer.

**Figure 50-9. Basic timer block diagram**




## Function overview

### Clock source

The basic timer can only be clocked by the 25MHz internal timer clock, which is from the source named CK_TIMER in RCU.

The CK_TIMER will be divided by 2500 to generate the 10kHz counter clock (PSC_CLK). The resolution of the counter is 100 µs.

### Down counting mode

When the CEN bit in TIMERx_CTL0 register is set, The basic timer loads the TIMER_CNT register with a pre-load value in the CPLOAD bit-filed of TIMER_CTL0 register. A new pre-load value can be written to the CPLOAD bit-filed by software at any time. If the CEN bit is set, the TIMER_CNT register will be immediately loaded to the new pre-load value and continue to count down from that value.

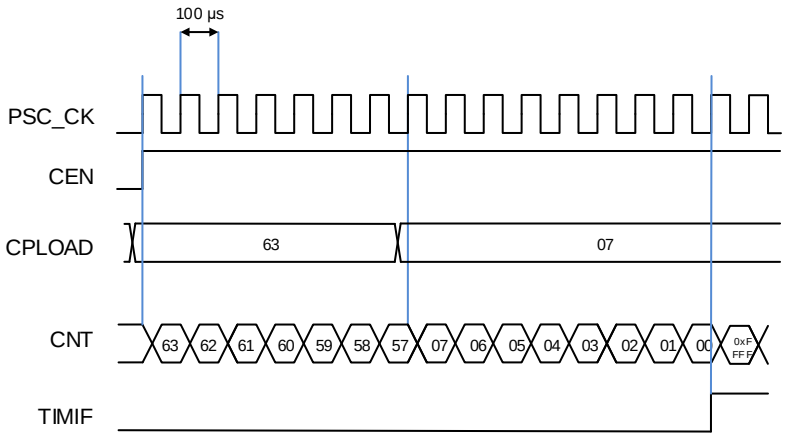
If a chip-level reset occurs or the CEN bit in TIMER_CTL0 register is converted from 1 to 0, The CPLOAD bit-filed is initialized to 0xFFFF. The CNT bit-filed is initialized to 0xFFFF on reset.

When the CEN bit in TIMER_CTL0 register is set, the counter counts down continuously from

the counter reload value to 0. Once the counter reaches 0, the counter wraps around to 0xFFFF and continues counting. The TIMIF bit in INTC_FLAG register is set. If TIMIE bit in INTC_EN register is set, basic timer generates the interrupt. If the TIMIF bit in INTC_FLAG register is set, it can only be cleared by writing a 1 to the bit.

The following figure shows a example of the counter behavior when CPLOAD bit-filed is converted from 0x63 to 0x07.

**Figure 50-10. Timing chart of down counting mode**




## Registers definition

Basic Timer base address: 0x0000 3800

### Control register 0 (TIMER_CTL0)

Address offset: 0x00
Reset value: 0x0000 FFFF

This register has to be accessed by word (32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | CEN | Reserved | | | | | | | | | | | | |
| | | rw | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPLOAD[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:30 | Reserved | Must be kept at reset value |
| 29 | CEN | Counter enable<br>0: Counter disable<br>1: Counter enable |
| 28:16 | Reserved | Must be kept at reset value |
| 15:0 | CPLOAD[15:0] | Counter pre-load value |

This bit-filed specifies the pre-load value of the counter.

When the CEN bit is set, the counter counts down from this value.

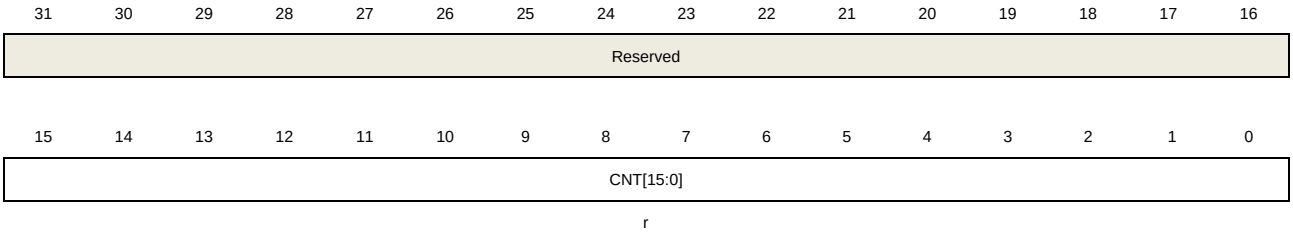
**Note:** When the CEN bit is converted from 1 to 0, this bit-filed wil be converted to 0xFFFF.

### Counter register (TIMER_CNT)

Address offset: 0x04

Reset value: 0x0000 FFFF

This register has to be accessed by word (32-bit).




| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | Reserved | Must be kept at reset value |
| 15:0 | CNT[15:0] | This bit-filed indicates the current counter value. |

## 50.6.2. Free-Running Counter (FRC)

### Overview

The Free-Running Counter has a 32-bit counter that can be used as an unsigned counter.The counter clock is 25MHz.

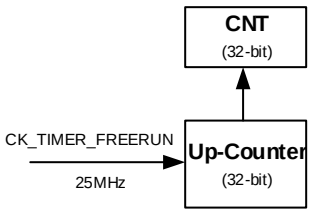
### Characteristics

- Counter width: 32 bits.
- Source of count clock is internal clock only.
- Counter mode: count up.

### Block diagram

*Figure 50-11. FRC block diagram* provides details on the internal configuration of the FRC.

**Figure 50-11. FRC block diagram**




### Function overview

#### Clock source

The FRC can only be clocked by the 25MHz internal timer clock, which is from the source named CK_TIMER_FREERUN in RCU.

#### Up counting mode

The counter counts up continuously from 0 to 0xFFFFFFFF and the counter frequency is 25MHz. Once the counter reaches the maximum, the counter recounts from 0. The FRC does not generate interrupts. If a chip-level reset occurs, the counter is initialized to 0.

The current count value can be read from FRC_CNT register.

The counter can take up to 160 ns to clear after a reset event.

### Registers definition

Free-Running Counter base address: 0x0000 3808

#### Counter Register (FRC_CNT)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT [31:16] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

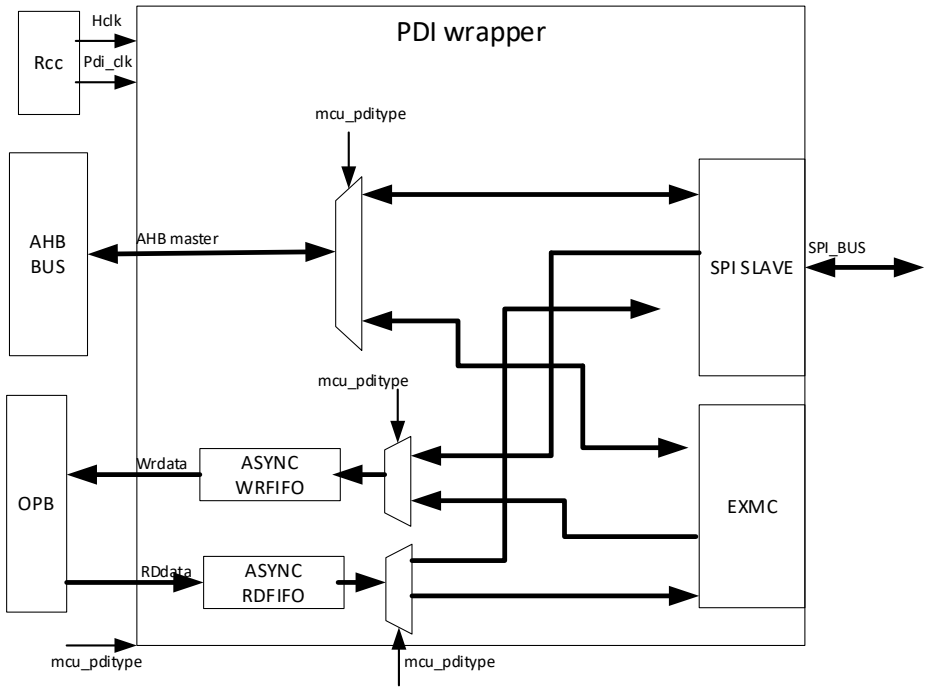
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT [15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | CNT[31:0] | This bit-filed indicates the current counter value.<br>If a chip-level reset occurs, the bit-filed is initialized to 0 and the counter counts up from 0.<br>Once the counter reaches the maximum, the bit-filed is converted to 0 and the counter continues to count up from 0.<br>**Note:** The counter can take up to 160 ns to clear after a reset event. |

## 50.7. PDI Wrapper

In the PDI wrapper, EXMC and SPI SLAVE are packaged into a wrapper for system integration. The PDI wrapper Used for data only selection SPI. The SPI SLAVE have only work, which is selected by pad of MCU_PDITYPE.

**Figure 50-12. Block diagram of PDI wrapper**




SPI SLAVE accesses registers through AHB channel and accesses CORE ram data through ASYNC RDFIFO and ASYNC WRFIFO.

When the pad of MCU_PDITYPE is 0, only the SPI SLAVE can access internal data.

PDI_CLK provides the clock for SPI_SLAVE, and ASYNC_FIFO. When the pad of MCU_PDITYPE is 0, PDI_CLK comes from SPI_SCK. HCLK is a 100MHz system clock that provides clocks for ASYNC_FIFO and SPI_SLAVE.

### 50.7.1. SPI/QSPI/OSPI slave

#### Overview

The EtherCAT support SPI / QSPI / OSPI slave module.

#### Characteristics

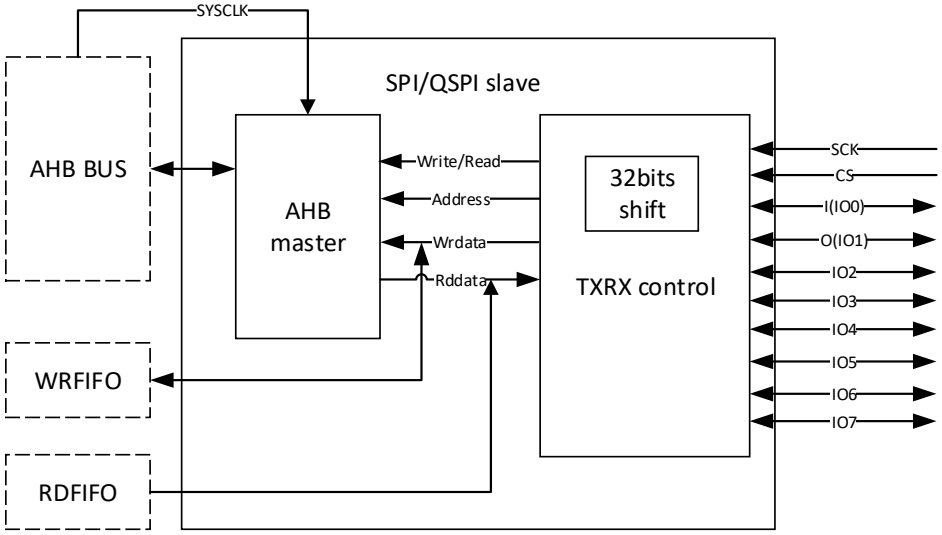
- Supports a maximum SPI clock rate of 75MHz.
- Only support slave mode.
- All samples are sampled along the rising edge.

- Supports FIFO buffer access.

## Block diagram

The block diagram of SPI is shown in ***Figure 50-13. Block diagram of SPI***.

**Figure 50-13. Block diagram of SPI**




## SPI signal description

### Pin description

The SPI/QSPI slave module contain 2 kinds of pin mode: 4-wire mode and 6-wire mode. All modes contain common pins, SCK and CS.

**Table 50-5. 4-wire mode**

| Pin name | Description |
|---|---|
| SCK | SPI CLK |
| CS | Slave select signal |
| I | Input pin, receive SPI/QSPI master data |
| O | Output pin, transmit data to SPI/QSPI master data |

**Table 50-6. 6-wire mode**

| Pin name | Description |
|---|---|
| SCK | SPI/QSPI CLK |
| CS | Slave select signal |
| SIO0 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO1 | Inout pin, receive SPI/QSPI master |

| | |
|---|---|
| | data and transmit data to SPI/QSPI master data |
| SIO2 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO3 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |

**Table 50-7. OSPI 8-line mode**

| Pin name | Description |
|---|---|
| SCK | Spi/qspi clk |
| CS | Slave select signal |
| SIO0 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO1 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO2 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO3 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO4 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO5 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO6 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |
| SIO7 | Inout pin, receive SPI/QSPI master data and transmit data to SPI/QSPI master data |

## SPI/QSPI/OSPI slave controller

### Function overview

The SPI slave interface can access registers and FIFOs with fewer pins. Single, Dual and Quad bit lanes are supported in SPI mode with a clock rate of up to 75MHz. QSPI mode always uses four bit lanes and also operates at up to 75 MHz.OSPI mode always uses eight bit lanes and also operates at up to 75MHz.

### Function description

The following is an overview of the functions provided by the SPI/ QSPI/OSPI Client:

- Fast Read: 4-wire (clock, select, data in and data out) reads at up to 75MHz. Serial command, address and data. Dummy byte(s) for first access. Single and multiple register reads with incrementing, decrementing or static addressing.

- Dual / Quad Output Read: 4 or 6-wire (clock, select, data in / out) reads at up to 75MHz. Serial command and address, parallel data. Dummy byte(s) for first access. Single and multiple register reads with incrementing, decrementing or static addressing.
- Dual / Quad I/O Read: 4 or 6-wire (clock, select, data in / out) reads at up to 75MHz. Serial command, parallel address and data. Dummy byte(s) for first access. Single and multiple register reads with incrementing, decrementing or static addressing.
- QSPI Read: 6-wire (clock, select, data in / out) writes at up to 75MHz. Parallel command, address and data. Dummy byte(s) for first access. Single and multiple register reads with incrementing, decrementing or static addressing.
- OSPI Read: 10-wire (clock, select, data in / out) writes at up to 75MHz. Parallel command, address and data. Dummy byte(s) for first access. Single and multiple register reads with incrementing, decrementing or static addressing.
- Write: 4-wire (clock, select, data in and data out) writes at up to 75MHz. Serial command, address and data. Single and multiple register writes with incrementing, decrementing or static addressing.
- Dual / Quad Data Write: 4 or 6-wire (clock, select, data in / out) writes at up to 75MHz. Serial command and address, parallel data. Single and multiple register writes with incrementing, decrementing or static addressing.
- Dual / Quad Address / Data Write: 4 or 6-wire (clock, select, data in / out) writes at up to 75MHz. Serial command, parallel address and data. Single and multiple register writes with incrementing, decrementing or static addressing.
- QSPI Write: 6-wire (clock, select, data in / out) writes at up to 75MHz. Parallel command, address and data. Single and multiple register writes with incrementing, decrementing or static addressing.
- OSPI Write: 10-wire (clock, select, data in / out) writes at up to 75MHz. Parallel command, address and data. Single and multiple register writes with incrementing, decrementing or static addressing.

**Operation description**

Input data on the IO [7:0] pins is sampled on the rising edge of the SCK input clock. Output data is sourced on the IO [7:0] pins with the falling edge of the clock. The SCK input clock can be either an active high pulse or an active low pulse. When the SCS# chip select input is high, the IO [7:0] inputs are ignored and the IO [7:0] outputs are three stated.

In SPI mode, the 8-bit instruction is started on the first rising edge of the input clock after SCS# goes active. The instruction is always input serially on I / IO0.

For read and write instructions, two address bytes follow the instruction byte. Depending on the instruction, the address bytes are input either serially, or 2 or 4 bits per clock. Although all registers are accessed as DWORDs, the address field is considered a byte address. bits 15 and 14 of the address field specifies that address is auto-decremented (10b) or auto-incremented (01b) for continuous accesses. (if accessing inner fifo, bits 15 and 14 will be ignored)

For all read instructions, dummy byte cycles follow the address bytes. The device does not drive the outputs during the dummy byte cycles. The dummy byte(s) are input either serially,

or 2 or 4 or 8 bits per clock. The data is input either serially, or 2 or 4 bits per clock.

For read and write instructions, one or more 32-bit data fields follow the dummy bytes (if present, else they follow the address bytes). The data is input either serially, or 2 or 4 or 8 bits per clock.

QSPI mode is entered from SPI with the Enable Quad I/O (EQIO) instruction. Once in QSPI mode, all further command, addresses, dummy bytes and data bytes are 4 bits per clock. QSPI mode can be exited using the Reset Quad I/O (RSTQIO) instruction.

OSPI mode is entered from SPI with the Enable Octa I/O (EOIO) instruction. Once in OSPI mode, all further command, addresses, dummy bytes and data bytes are 8 bits per clock. OSPI mode can be exited using the Reset Octa I/O (RSTQIO) instruction

All instructions, addresses and data are transferred with the most-significant bit (msb) or di-bit (msd) or nibble (msn) first. Addresses are transferred with the most-significant byte (MSB) first. Data is transferred with the least-significant byte (LSB) first (little endian).

The SPI interface supports up to a 75MHz input clock.

The SPI interface supports a minimum time of 50ns between successive commands (a minimum SCS# inactive time of 50 ns).

The instructions supported in SPI mode are listed in ***Table 50-8. SPI instructions***. QSPI instructions are listed in ***Table 50-9. QSPI instruction***. OSPI instructions are listed in ***Table 50-10. OSPI instruction***. Unsupported instructions are must not be used.

**Table 50-8. SPI instructions**

| Instruction | Description | Bit width | Inst code | Addr Bytes | Dummy Bytes | Data bytes | Max Freq |
|---|---|---|---|---|---|---|---|
| Configuration | | | | | | | |
| EQIO | Enable QSPI | 1-0-0 | 38h | 0 | 0 | 0 | 75MHz |
| EOIO | Enable OSPI | 1-0-0 | 3Ah | 0 | 0 | 0 | 75MHz |
| RSTIO | Reset SPI | 1-0-0 | FFh | 0 | 0 | 0 | 75MHz |
| Read | | | | | | | |
| READ | Read | 1-1-1 | 0Bh | 2 | 1 | 4 to ∞ | 75MHz |
| SDOR | SPI Dual Output Read | 1-1-2 | 3Bh | 2 | 1 | 4 to ∞ | 75MHz |
| SDIOR | SPI Dual I/O Read | 1-2-2 | BBh | 2 | 2 | 4 to ∞ | 75MHz |
| SQOR | SPI Quad Output Read | 1-1-4 | 6Bh | 2 | 1 | 4 to ∞ | 75MHz |
| SQIOR | SPI Quad I/O Read | 1-4-4 | EBh | 2 | 4 | 4 to ∞ | 75MHz |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Write | | | | | | | |
| WRITE | Write | 1-1-1 | 02h | 2 | 0 | 4 to ∞ | 75MHz |
| SDDW | SPI Dual Data Write | 1-1-2 | 32h | 2 | 0 | 4 to ∞ | 75MHz |
| SDADW | SPI Dual Address / Data Write | 1-2-2 | B2h | 2 | 0 | 4 to ∞ | 75MHz |
| SQDW | SPI Quad Data Write | 1-1-4 | 62h | 2 | 0 | 4 to ∞ | 75MHz |
| SQADW | SPI Quad Address / Data Write | 1-4-4 | E2h | 2 | 0 | 4 to ∞ | 75MHz |

**Table 50-9. QSPI instruction**

| Instruction | Description | Bit width | Inst code | Addr Bytes | Dummy Bytes | Data bytes | Max Freq |
|---|---|---|---|---|---|---|---|
| Configuration | | | | | | | |
| RSTQIO | Reset QSPI | 4-0-0 | FFh | 0 | 0 | 4 to ∞ | 75MHz |
| Read | | | | | | | |
| READ | Read | 4-4-4 | 0Bh | 2 | 3 | 4 to ∞ | 75MHz |
| Write | | | | | | | |
| WRITE | write | 4-4-4 | 02h | 2 | 0 | 4 to ∞ | 75MHz |

**Table 50-10. OSPI instruction**

| Instruction | Description | Bit width | Inst code | Addr Bytes | Dummy Bytes | Data bytes | Max Freq |
|---|---|---|---|---|---|---|---|
| Configuration | | | | | | | |
| RSTOIO | Reset OSPI | 8-0-0 | FFh | 0 | 0 | 4 to ∞ | 75MHz |
| Read | | | | | | | |
| READ | Read | 8-8-8 | 0Bh | 2 | 8 | 4 to ∞ | 75MHz |
| Write | | | | | | | |
| WRITE | write | 8-8-8 | 02h | 2 | 0 | 4 to ∞ | 75MHz |

**Note:** The bit width format is: command bit width, address / dummy bit width, data bit width. For example, 1-2-4 means command uses 1 line, address/dummy uses 2 lines, data uses 4 lines.

### SPI configuration commands

#### Enable QSPI

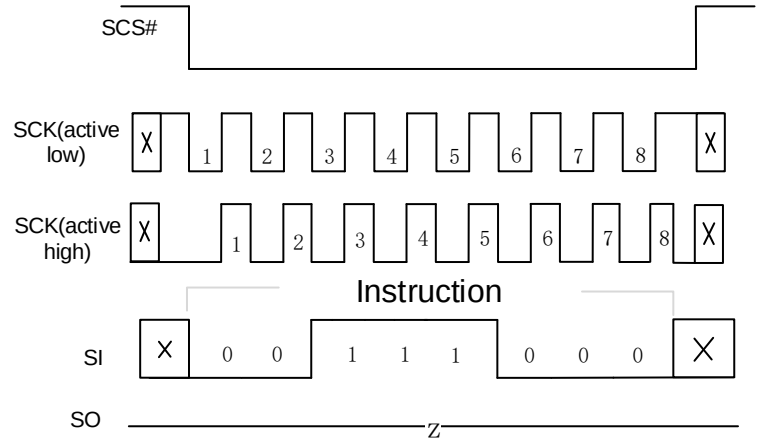
The enable QSPI instruction changes the mode of operation to QSPI. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

The SPI client interface is selected by first bringing SCS# active. The 8-bit EQIO instruction, 38h, is input into the I / IO [0] pin one bit per clock. The SCS# input is brought inactive to

conclude the cycle.

*Figure 50-14. Enable QSPI* illustrates the Enable QSPI instruction.

**Figure 50-14. Enable QSPI**



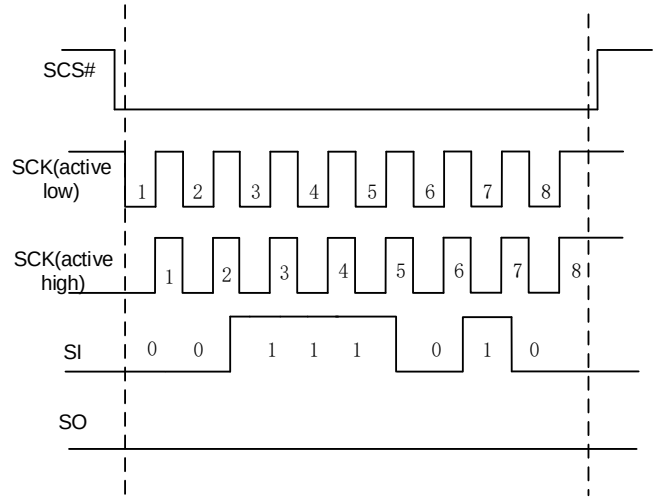

**Enable OSPI**

The Enable OSPI instruction changes the mode of operation to OSPI. This instruction is supported in SPI bus protocol only with clock frequencies up to 75 MHz. This instruction is not supported in OSPI bus protocol.

The SPI client interface is selected by first bringing SCS# active. The 8-bit EOIO instruction, 3Ah, is input into the I / IO[0] pin one bit per clock. The SCS# input is brought inactive to conclude the cycle.

*Figure 50-15. Enable OSPI* illustrates the Enable QSPI instruction.

**Figure 50-15. Enable OSPI**




**Reset QSPI**

The Reset QSPI/OSPI instruction changes the mode of operation to SPI. This instruction is supported in SPI/QSPI/OSPI bus protocols with clock frequencies up to 75MHz.

The SPI/QSPI/OSPI client interface is selected by first bringing SCS# active. The 8-bit RSTQIO instruction, FFh, is input into the I / IO[0] pin, one bit per clock, in SPI mode and into the IO[3:0] pins, four bits per clock, in QSPI mode. The SCS# input is brought inactive to conclude the cycle.

***Figure 50-16. SPI MODE RESET SPI*** illustrates the Reset SPI instruction for SPI mode.

***Figure 50-17. QSPI MODE RESET QSPI*** illustrates the Reset QSPI instruction for QSPI mode.

***Figure 50-18. OSPI MODE RESET OSPI*** illustrates the Reset OSPI instruction for OSPI mode.

**Figure 50-16. SPI MODE RESET SPI**

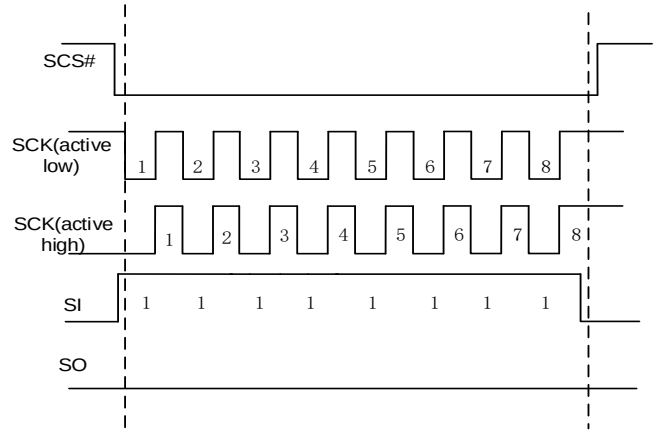



**Figure 50-17. QSPI MODE RESET QSPI**

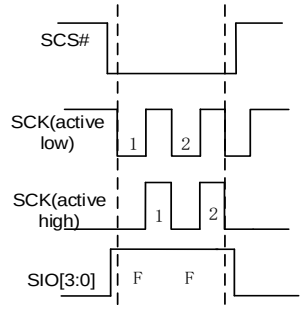
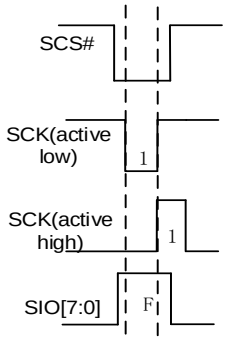



**Figure 50-18. OSPI MODE RESET OSPI**



### SPI READ COMMANDS

Various read commands are support by the SPI / QSPI client. The following applies to all read commands.

#### MULTIPLE READS

Additional reads, beyond the first, are performed by continuing the clock pulses while SCS# is active. The upper two bits of the address specify auto-incrementing (address[15:14]=01b) or auto-decrementing (address[15:14]=10b). The internal DWORD address is incremented, decremented, or maintained based on these bits. Maintaining a fixed internal address is useful for register polling.

#### READ

The Read instruction inputs the instruction code and the address and dummy bytes one bit per clock and outputs the data one bit per clock. In QSPI mode, the instruction code and the address and dummy bytes are input four bits per clock and the data is output four bits per clock. This instruction is supported in SPI and QSPI bus protocols with clock frequencies up to 75MHz.

The SPI/QSPI/OSPI client interface is selected by first bringing SCS# active. For SPI mode, the 8-bitREAD instruction, 0Bh, is input into the I / IO [0] pin, followed by the two address bytes and 1 dummy byte. For QSPI mode, the 8-bit FASTREAD instruction is input into the IO [3:0] pins, followed by the two address bytes and 3 dummy bytes. The address bytes specify a BYTE address within the device. For OSPI mode, the 8-bit FASTREAD instruction is input into the IO [7:0] pins, followed by the two address bytes and 8 dummy bytes. The address bytes specify a BYTE address within the device.

On the falling clock edge following the rising edge of the last dummy bit (or nibble), the O / IO [1] pin is driven starting with the msb of the LSB of the selected register. For QSPI mode, IO [3:0] are driven starting with the msn of the LSB of the selected register. For OSPI mode, IO [7:0] are driven starting with the msn of the LSB of the selected register. The remaining register bits are shifted out on subsequent falling clock edges.

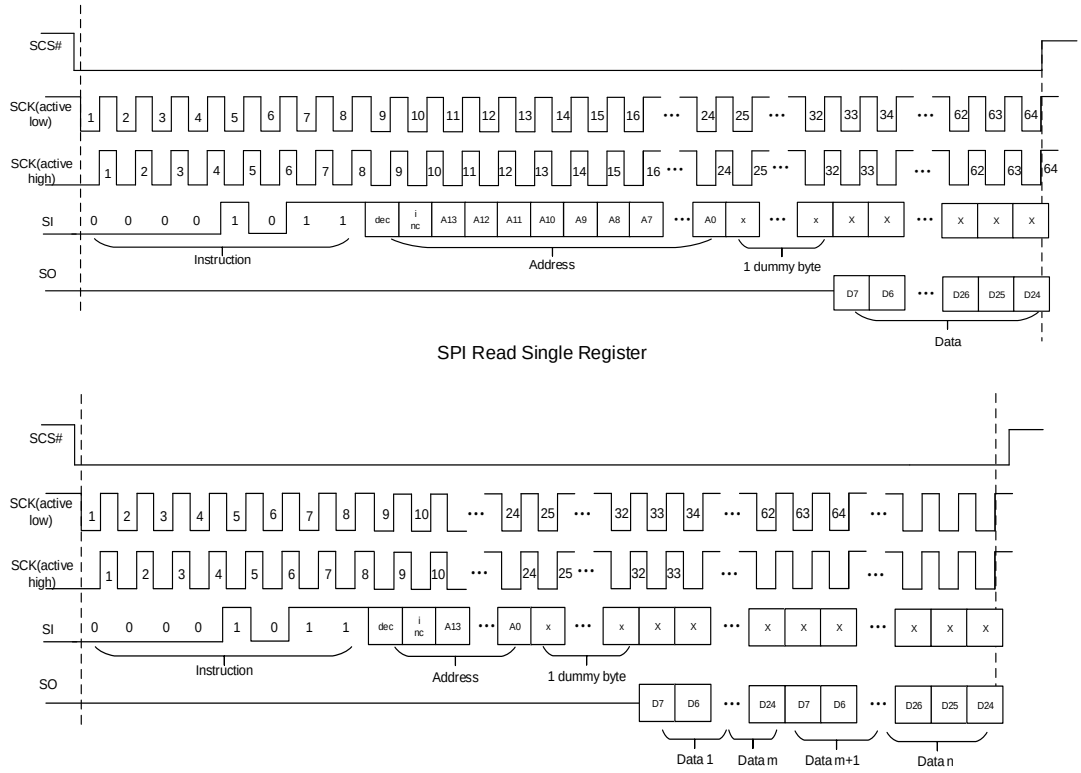
The SCS# input is brought inactive to conclude the cycle. The O / IO [7:0] pins are three-stated at this time.

***Figure 50-19. SPI READ*** illustrates a typical single and multiple register fast read for SPI mode.

***Figure 50-20. QSPI READ*** illustrates a typical single and multiple register fast read for QSPI mode.

***Figure 50-21. OSPI READ*** illustrates a typical single and multiple register fast read for OSPI mode.

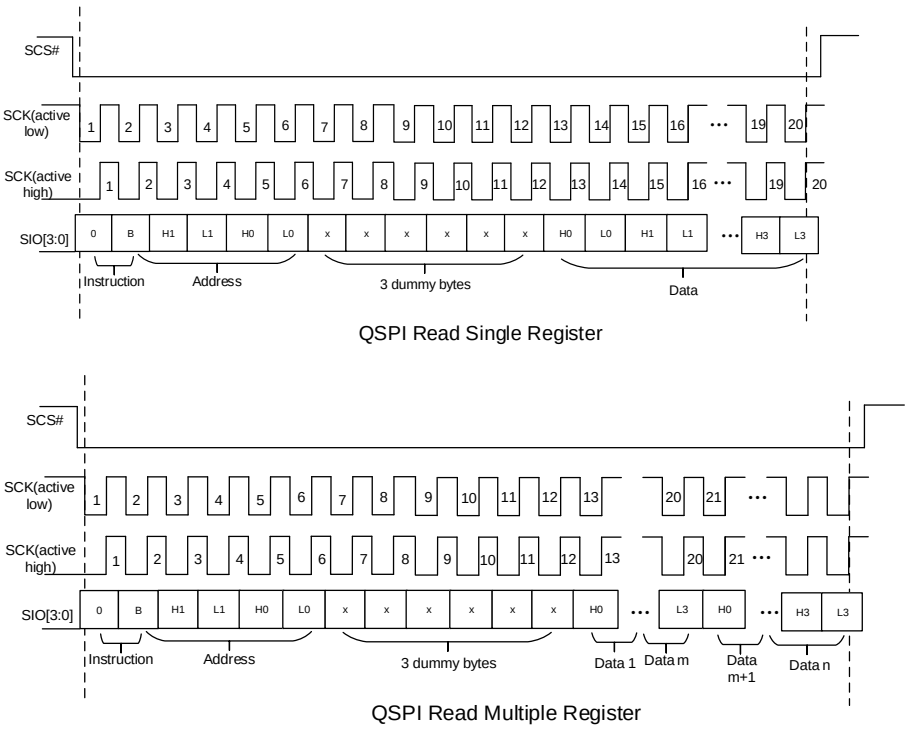
**Figure 50-19. SPI READ**




SPI Read Single Register

SPI Read Multiple Register

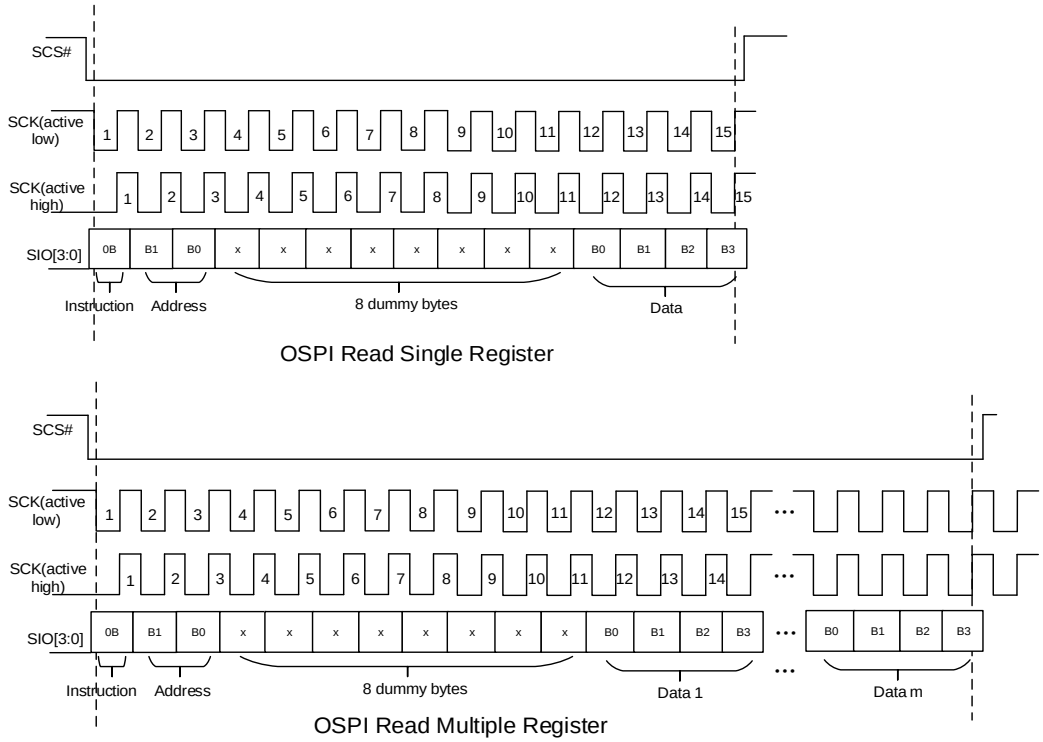
**Figure 50-20. QSPI READ**




QSPI Read Single Register

QSPI Read Multiple Register

**Figure 50-21. OSPI READ**




**Dual Output Read**

The SPI Dual Output Read instruction inputs the instruction code and the address and dummy bytes one bit per clock and outputs the data two bits per clock. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

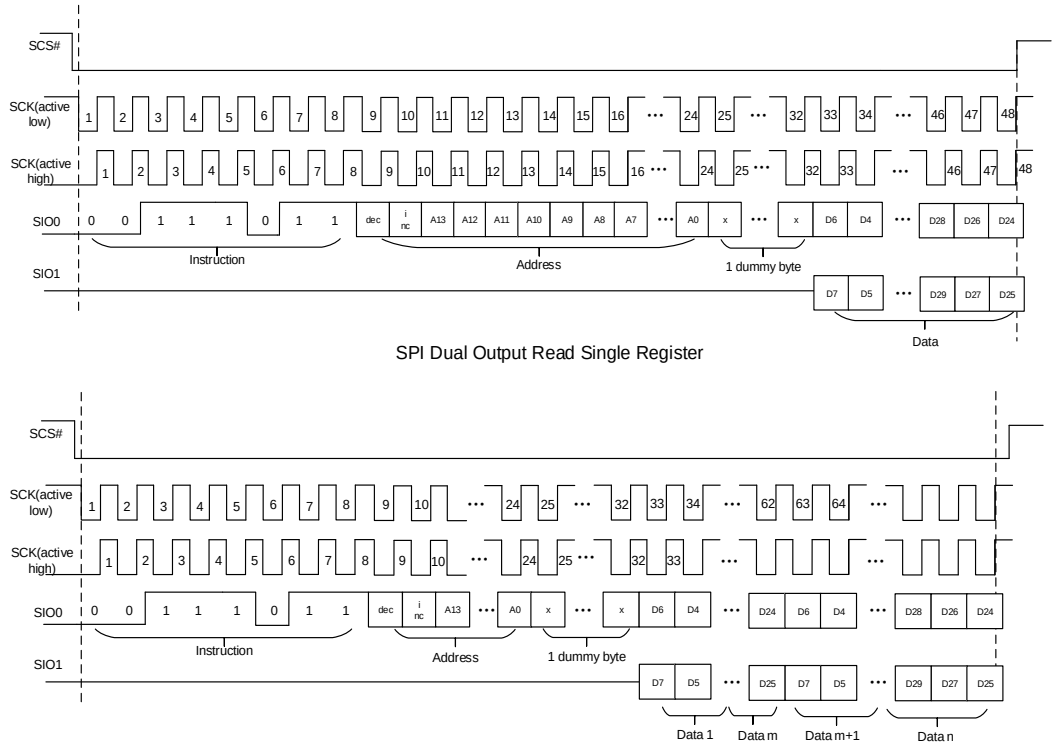
The SPI client interface is selected by first bringing SCS# active. The 8-bit SDOR instruction, 3Bh, is input into the IO [0] pin, followed by the two address bytes and 1 dummy byte. The address bytes specify a BYTE address within the device.

On the falling clock edge following the rising edge of the last dummy di-bit, the IO [1:0] pins are driven starting with the msbs of the LSB of the selected register. The remaining register di-bits are shifted out on subsequent falling clock edges.

The SCS# input is brought inactive to conclude the cycle. The IO [1:0] pins are three-stated at this time.

***Figure 50-22. SPI DUAL OUTPUT READ*** illustrates a typical single and multiple register dual output read.

**Figure 50-22. SPI DUAL OUTPUT READ**




**QUAD Output Read**

The SPI Quad Output Read instruction inputs the instruction code and the address and dummy bytes one bit per clock and outputs the data four bits per clock. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

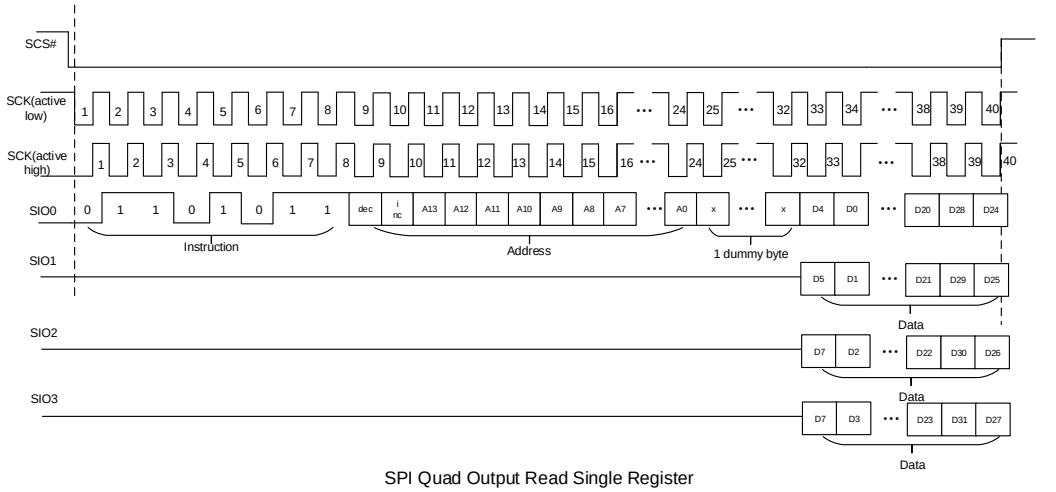
The SPI client interface is selected by first bringing SCS# active. The 8-bit SQOR instruction, 6Bh, is input into the IO [0] pin, followed by the two address bytes and 1 dummy byte. The address bytes specify a BYTE address within the device.

On the falling clock edge following the rising edge of the last dummy bit, the IO [3:0] pins are driven starting with the msn of the LSB of the selected register. The remaining register nibbles are shifted out.

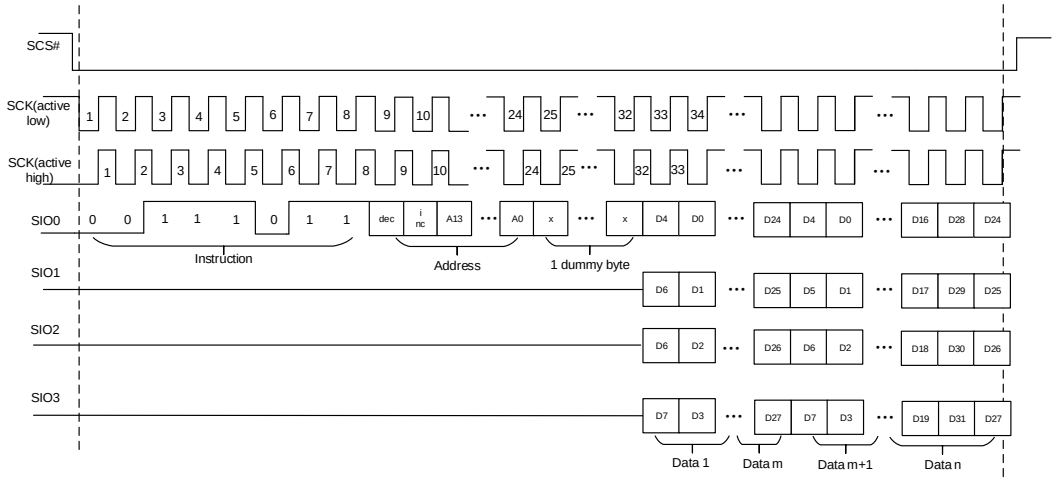
The SCS# input is brought inactive to conclude the cycle. The IO [3:0] pins are three-stated at this time.

***Figure 50-23. SPI QUAD OUTPUT READ*** illustrates a typical single and multiple register quad output read.

**Figure 50-23. SPI QUAD OUTPUT READ**




SPI Quad Output Read Single Register




SPI Quad Output Read Multiple Register

**Dual I/O Read**

The SPI Dual I/O Read instruction inputs the instruction code one bit per clock and the address and dummy bytes two bits per clock and outputs the data two bits per clock. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

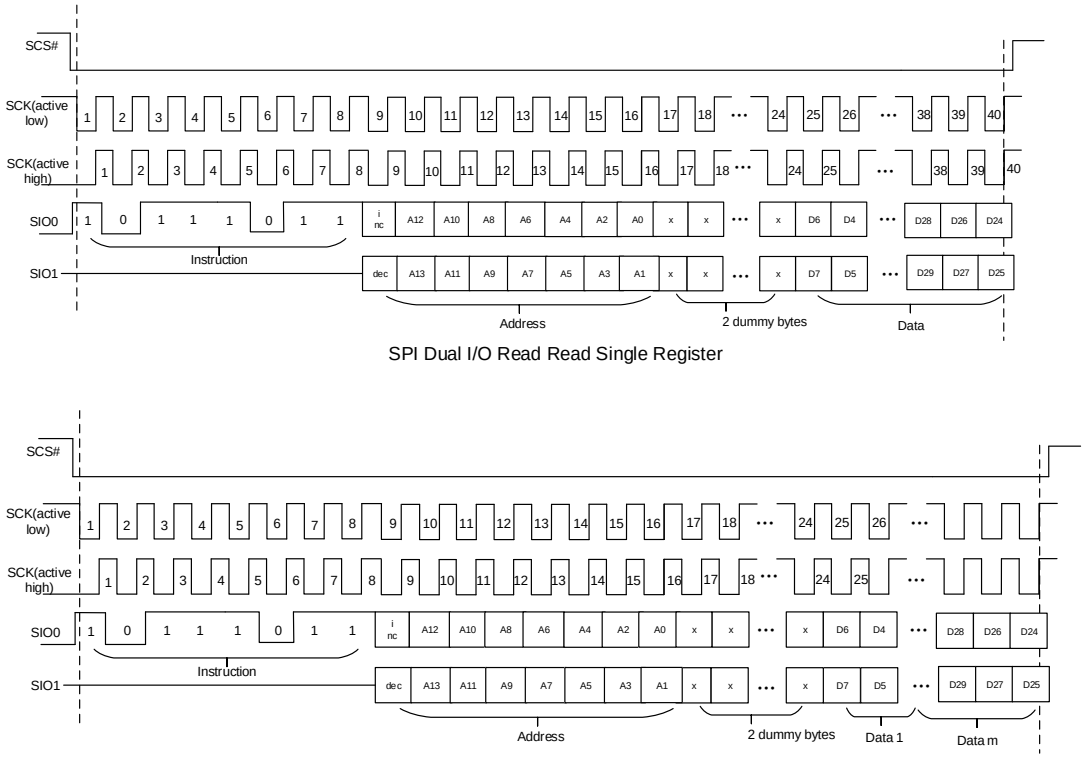
The SPI client interface is selected by first bringing SCS# active. The 8-bit SDIOR instruction, BBh, is input into the IO [0] pin, followed by the two address bytes and 2 dummy bytes into the IO [1:0] pins. The address bytes specify a BYTE address within the device.

On the falling clock edge following the rising edge of the last dummy di-bit, the IO [1:0] pins are driven starting with the msbs of the LSB of the selected register. The remaining register di-bits are shifted out on subsequent falling clock edges.

The SCS# input is brought inactive to conclude the cycle. The IO [1:0] pins are three-stated at this time.

***Figure 50-24. SPI DUAL I/O READ*** illustrates a typical single and multiple register dual I/O read.

**Figure 50-24. SPI DUAL I/O READ**




**Quad I/O Read**

The SPI Quad I/O Read instruction inputs the instruction code one bit per clock and the address and dummy bytes four bits per clock and outputs the data four bits per clock. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

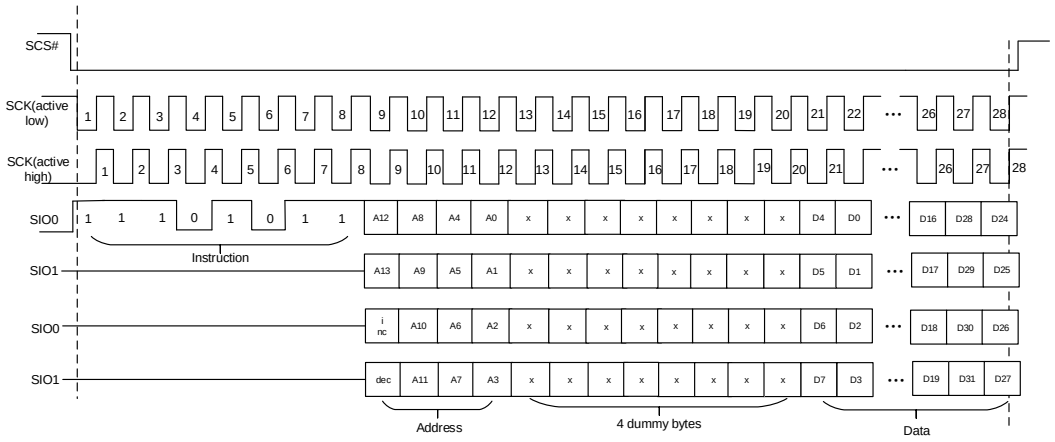
The SPI client interface is selected by first bringing SCS# active. The 8-bit SQIOR instruction, EBh, is input into the IO [0] pin, followed by the two address bytes and 4 dummy bytes into the IO [3:0] pins. The address bytes specify a BYTE address within the device.

On the falling clock edge following the rising edge of the last dummy nibble, the IO [3:0] pins are driven starting with the msn of the LSB of the selected register. The remaining register nibbles are shifted out on subsequent falling clock edges.

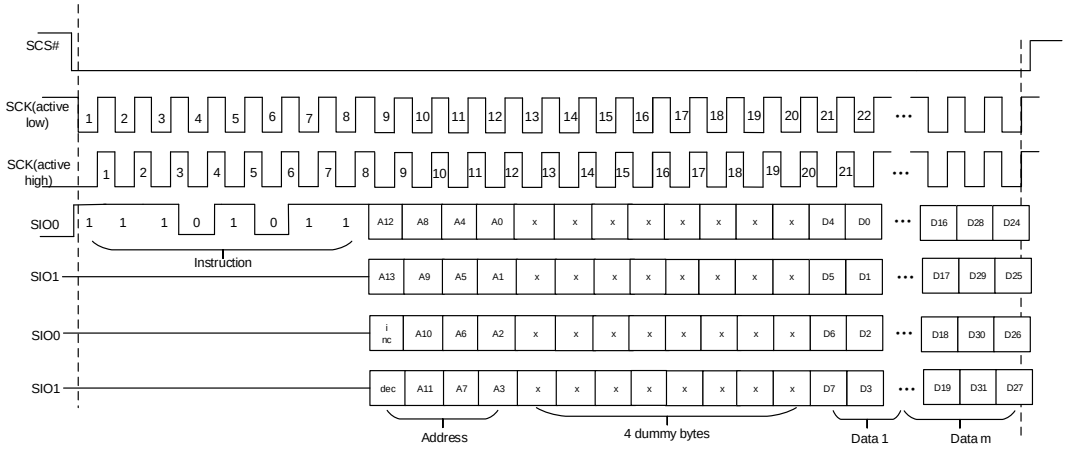
The SCS# input is brought inactive to conclude the cycle. The IO [3:0] pins are three-stated at this time.

***Figure 50-25. SPI QUAD I/O READ*** illustrates a typical single and multiple register dual output read.

**Figure 50-25. SPI QUAD I/O READ**




SPI Qual I/O Read Read Single Register




SPI Quad I/O Read Multiple Register

## SPI WRITE COMMANDS

Multiple write commands are support by the SPI/QSPI client. The following applies to all write commands.

### MULTIPLE WRITES

Multiple reads are performed by continuing the clock pulses and input data while SCS# is active. The upper two bits of the address specify auto-incrementing (address[15:14]=01b) or auto-decrementing (address[15:14]=10b). The internal DWORD address is incremented, decremented, or maintained based on these bits. Maintaining a fixed internal address may be useful for register “bit-banging” or other repeated writes.

#### Write

The Write instruction inputs the instruction code and address and data bytes one bit per clock. In QSPI mode, the instruction code and the address and data bytes are input four bits per clock. This instruction is supported in SPI and QSPI bus protocols with clock frequencies up to 75MHz.

The SPI/QSPI client interface is selected by first bringing SCS# active. For SPI mode, the 8-bit WRITE instruction, 02h, is input into the I / IO [0] pin, followed by the two address bytes. For QSPI mode, the 8-bit WRITE instruction, 02h, is input into the IO [3:0] pins, followed by the two address bytes. For OSPI mode, the 8-bit WRITE instruction, 02h, is input into the IO [7:0] pins, followed by the two address bytes. The address bytes specify a BYTE address within the device.

The data follows the address bytes. For SPI mode, the data is input into the I / IO [0] pin starting with the msb of the LSB. For QSPI mode the data is input nibble wide using IO [3:0] starting with the msn of the LSB. For OSPI mode the data is input nibble wide using IO [7:0] starting with the msn of the LSB. The remaining bits/ nibbles are shifted in on subsequent clock edges. The data write to the register occurs after the 32-bits are input. In the event that 32-bits are not written when the SCS# is returned high, the write is considered invalid and the register is not affected.

The SCS# input is brought inactive to conclude the cycle.

***Figure 50-26. SPI WRITE*** illustrates a typical single and multiple register write for SPI mode.

***Figure 50-27. QSPI WRITE*** illustrates a typical single and multiple register write for QSPI mode.

***Figure 50-28. OSPI WRITE*** illustrates a typical single and multiple register write for OSPI mode.

**Figure 50-26. SPI WRITE**

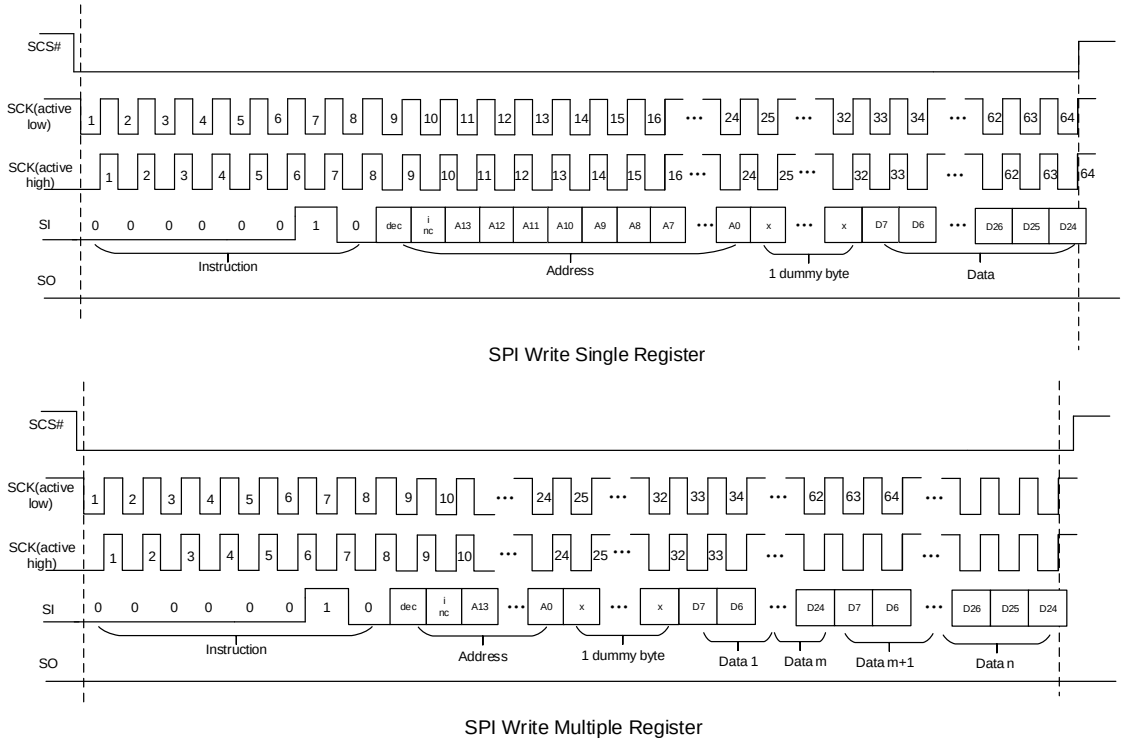


**Figure 50-27. QSPI WRITE**

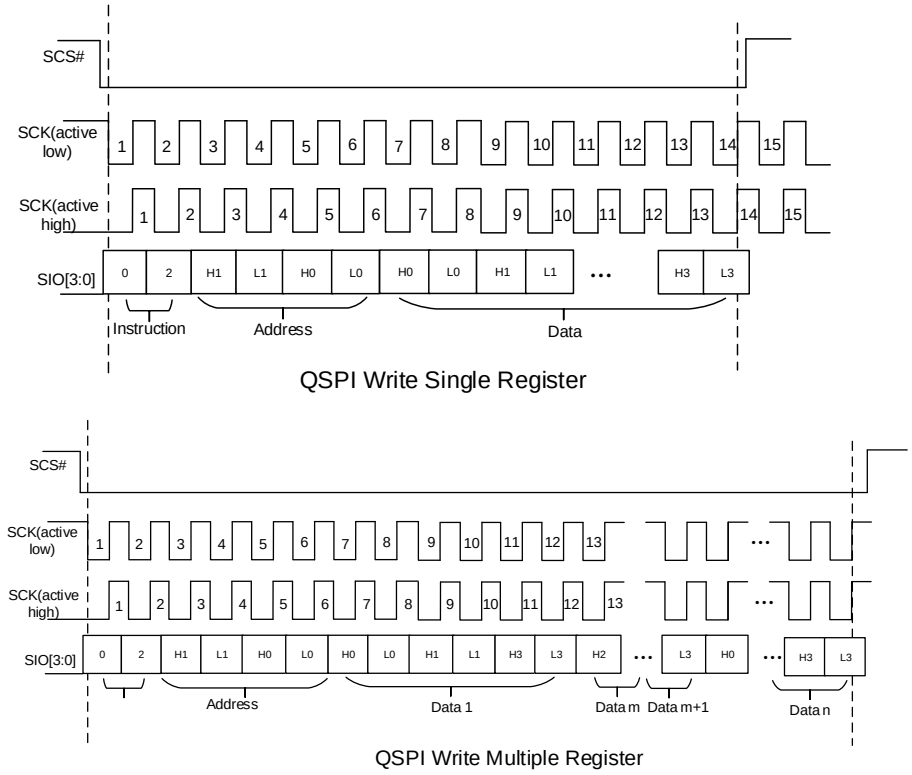
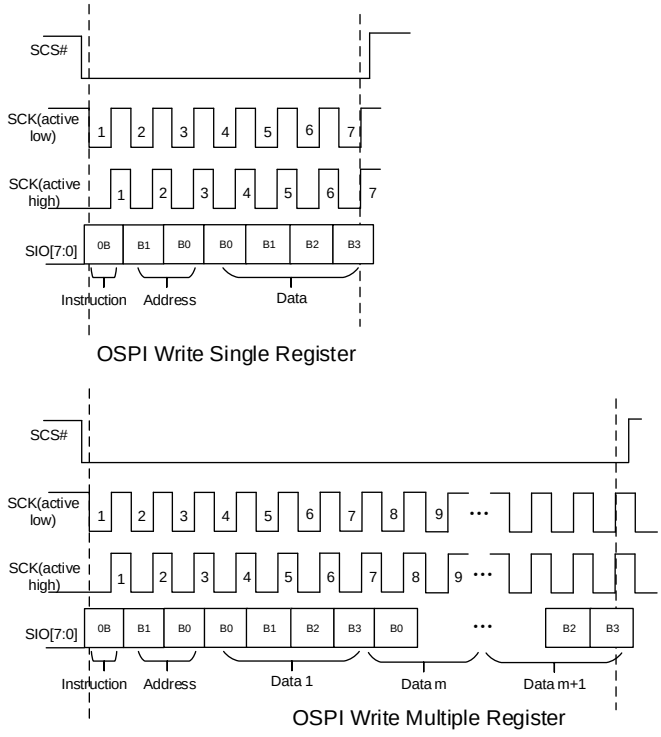



**Figure 50-28. OSPI WRITE**




**Dual Data Read**

The SPI Dual Data Write instruction inputs the instruction code and address bytes one bit per clock and inputs the data two bits per clock. This instruction is supported in SPI bus protocol

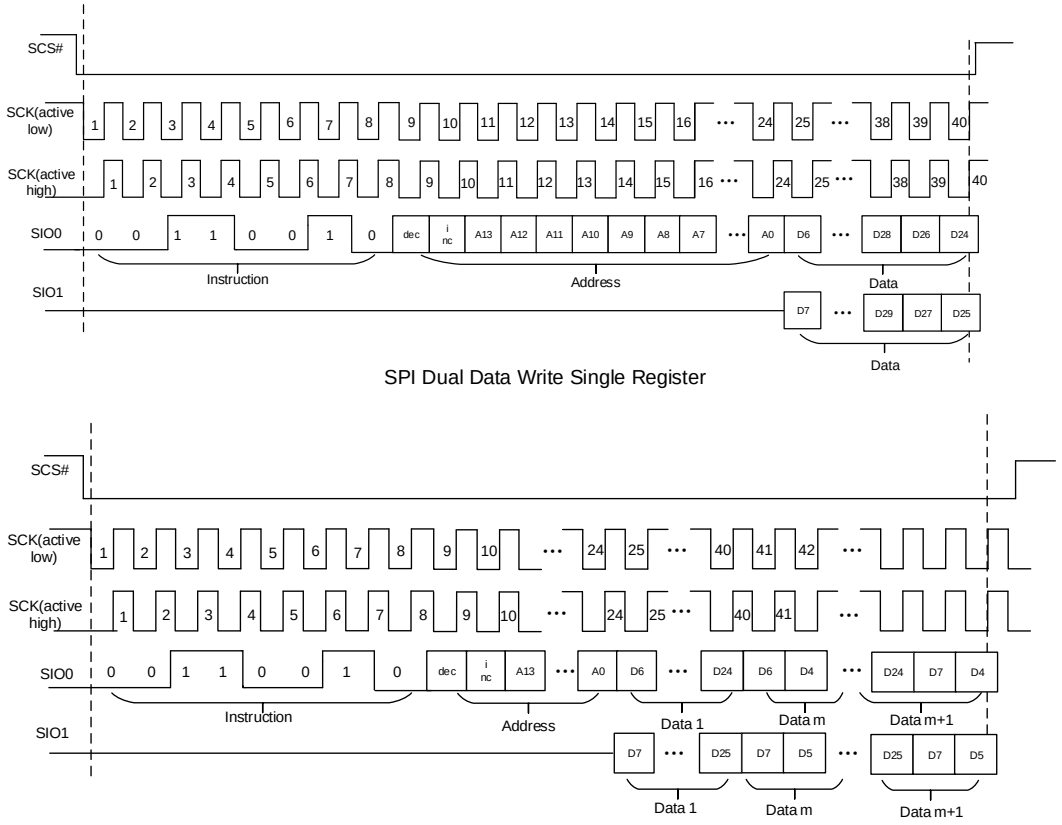
only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

The SPI client interface is selected by first bringing SCS# active. The 8-bit SDDW instruction, 32h, is input into the IO [0] pin, followed by the two address bytes. The address bytes specify a BYTE address within the device. The data follows the address bytes. The data is input into the IO [1:0] pins starting with the msbs of the LSB. The remaining di-bits are shifted in on subsequent clock edges. The data write to the register occurs after the 32-bits are input. In the event that 32-bits are not written when the SCS# is returned high, the write is considered invalid and the register is not affected.

The SCS# input is brought inactive to conclude the cycle.

*Figure 50-29. SPI DUAL DATA WRITE* illustrates a typical single and multiple register dual data write.

**Figure 50-29. SPI DUAL DATA WRITE**




**Quad Data Read**

The SPI Quad Data Write instruction inputs the instruction code and address bytes one bit per clock and inputs the data four bits per clock. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

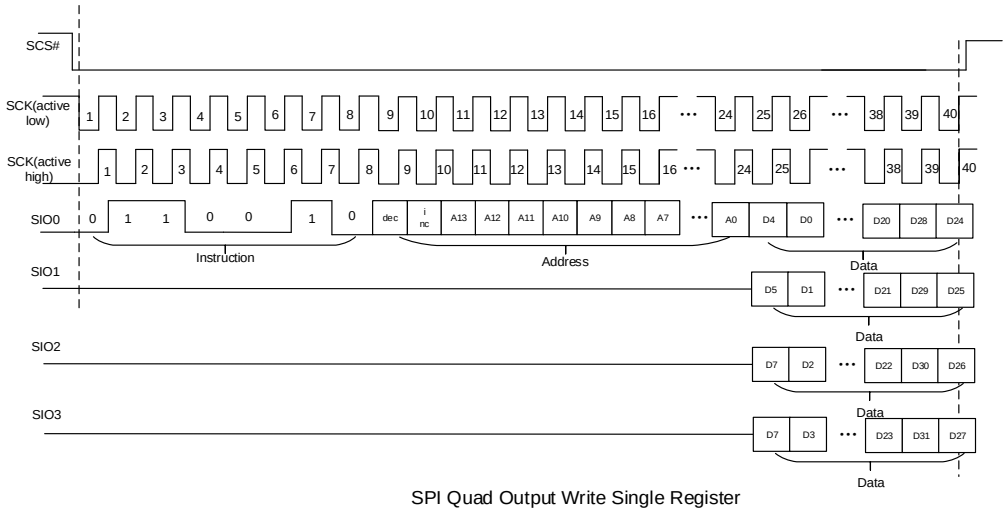
The SPI slave interface is selected by first bringing SCS# active. The 8-bit SQDW instruction, 62h, is input into the IO[0] pin, followed by the two address bytes. The address bytes specify a BYTE address within the device.

The data follows the address bytes. The data is input into the IO[3:0] pins starting with the msn of the LSB. The remaining nibbles are shifted in on subsequent clock edges. The data write to the register occurs after the 32-bits are input. In the event that 32-bits are not written when the SCS# is returned high, the write is considered invalid and the register is not affected.

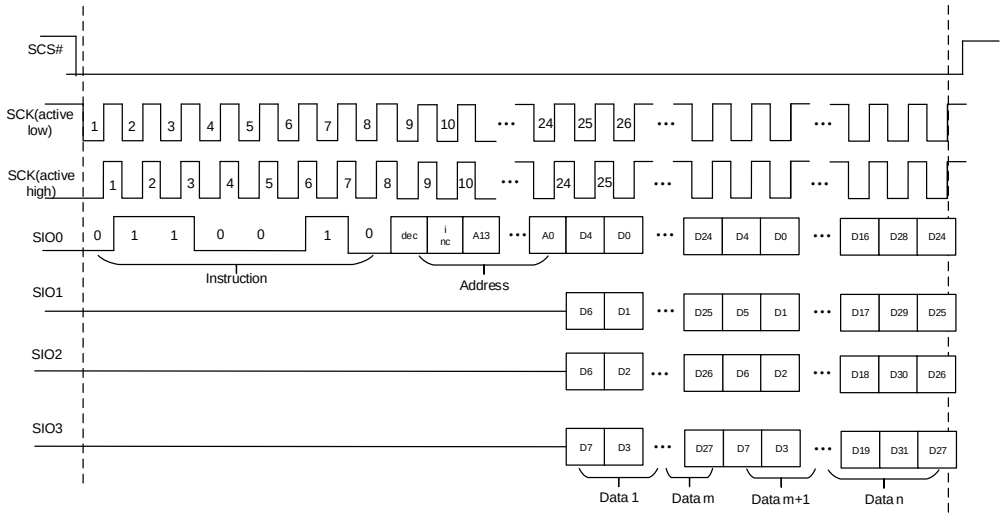
The SCS# input is brought inactive to conclude the cycle.

*Figure 50-30. SPI QUAD DATA WRITE* illustrates a typical single and multiple register quad data write.

**Figure 50-30. SPI QUAD DATA WRITE**




SPI Quad Output Write Single Register




SPI Quad Output Write Multiple Register

**Dual Address / Data Write**

The SPI Dual Address / Data Write instruction inputs the instruction code one bit per clock and the address and data bytes two bits per clock. This instruction is supported in SPI bus

protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

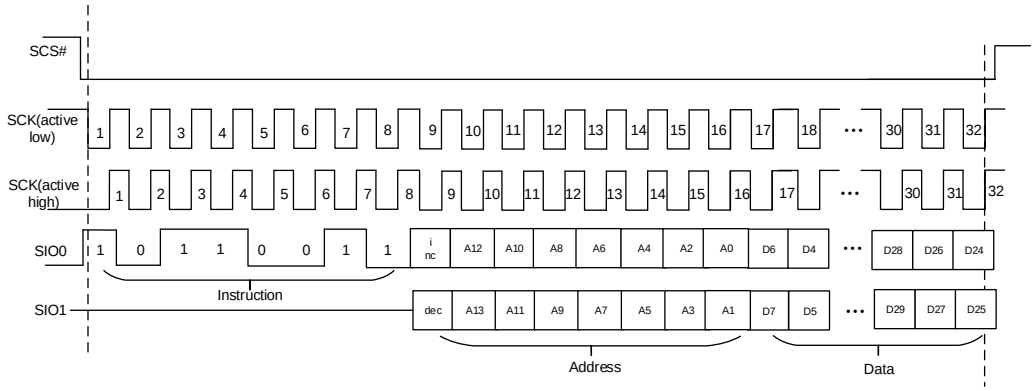
The SPI client interface is selected by first bringing SCS# active. The 8-bit SDADW instruction, B2h, is input into the IO[0] pin, followed by the two address bytes into the IO[1:0] pins. The address bytes specify a BYTE address within the device.

The data follows the address bytes. The data is input into the IO[1:0] pins starting with the msbs of the LSB. The remaining di-bits are shifted in on subsequent clock edges. The data write to the register occurs after the 32-bits are input. In the event that 32-bits are not written when the SCS# is returned high, the write is considered invalid and the register is not affected.

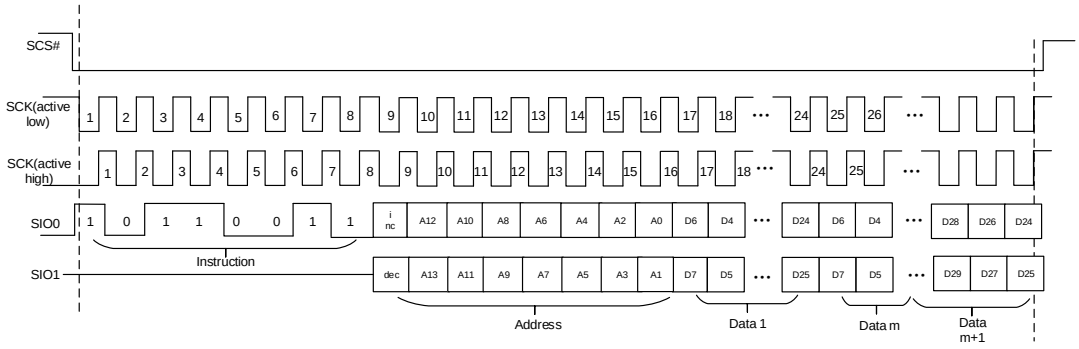
The SCS# input is brought inactive to conclude the cycle.

*Figure 50-31. SPI DUAL ADDRESS / DATA WRITE* illustrates a typical single and multiple register dual address / data write.

**Figure 50-31. SPI DUAL ADDRESS / DATA WRITE**




SPI Dual Address / Data Write Single Register




SPI Dual Address / Data Write Multiple Register

**Quad Address / Data Write**

The SPI Quad Address / Data Write instruction inputs the instruction code one bit per clock and the address and data bytes four bits per clock. This instruction is supported in SPI bus protocol only with clock frequencies up to 75MHz. This instruction is not supported in QSPI bus protocol.

The SPI client interface is selected by first bringing SCS# active. The 8-bit SQADW instruction,

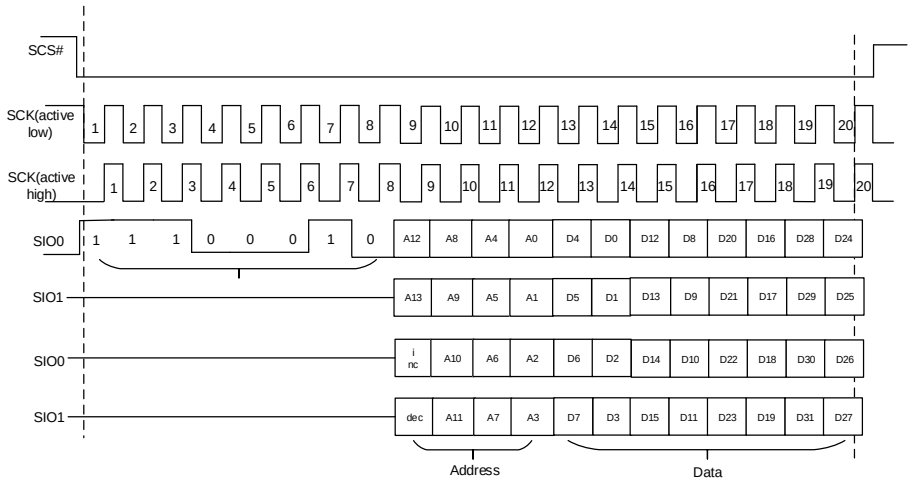
E2h, is input into the IO[0] pin, followed by the two address bytes into the IO[3:0] pins. The address bytes specify a BYTE address within the device.

The data follows the address bytes. The data is input into the IO[3:0] pins starting with the msn of the LSB. The remaining nibbles are shifted in on subsequent clock edges. The data write to the register occurs after the 32-bits are input. In the event that 32-bits are not written when the SCS# is returned high, the write is considered invalid and the register is not affected.

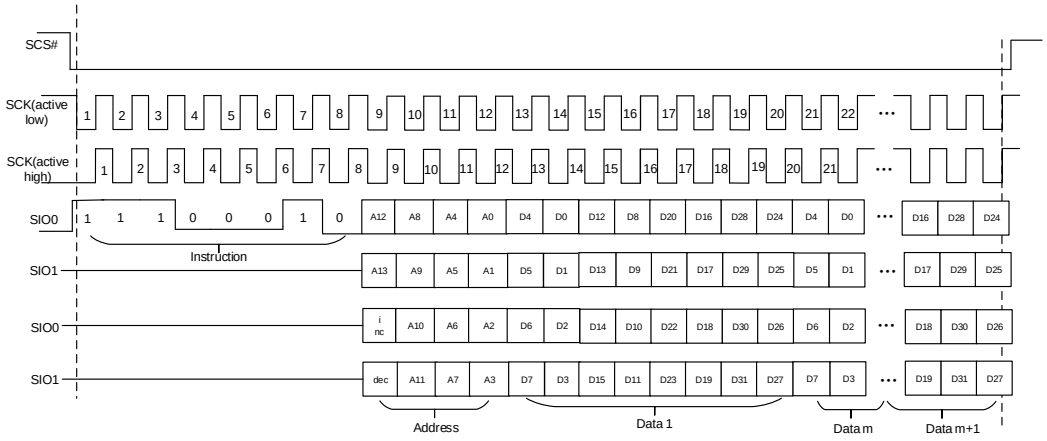
The SCS# input is brought inactive to conclude the cycle.

*Figure 50-32. SPI QUAD ADDRESS / DATA WRITE* illustrates a typical single and multiple register quad address / data write.

**Figure 50-32. SPI QUAD ADDRESS / DATA WRITE**




SPI Qual I/O Read Read Single Register




SPI Quad Address/Data Write Multiple Register

**SPI WAKE UP SYSTEM**

When Chip has entered Low Power mode, User can access ByteTest and READY register to exit Low Power Mode by SPI/QSPI/OSPI.

To determine when the host interface is functional, the Byte Order Test Register (BYTE_TEST) should be polled. Once the correct pattern is read, the interface can be considered functional.

At this point, the Device Ready (READY) register can be polled to determine when the device is fully awake.

**SPI ACCESS FIFO**

SPI/QSPI/OSPI supports access to registers and FIFO. In the process of accessing FIFO, users need to ensure that FIFO are not out of the boundary during access, otherwise data loss will occur.

The BUS module provides some FIFO Count registers, PRAM_RD_AVAIL_CNT, PRAM_WR_AVAIL_CNT, which can be read out through SPI bus. When the user reads TXFIFO, the number of read data should be less than or equal to PRAM_RD_AVAIL_CNT. When the user writes RXFIFO, the number of write data should be less than or equal to PRAM_WR_AVAIL_CNT.

When reading TXFIFO, it is recommended to read PRAM_RD_AVAIL_CNT first and then read PRAM_RD_AVAIL_CNT data in TXFIFO.

When writing RXFIFO, it is recommended to read PRAM_WR_AVAIL_CNT first, and write PRAM_WR_AVAIL_CNT data to RXFIFO.

## 50.8. Ethernet PHYS

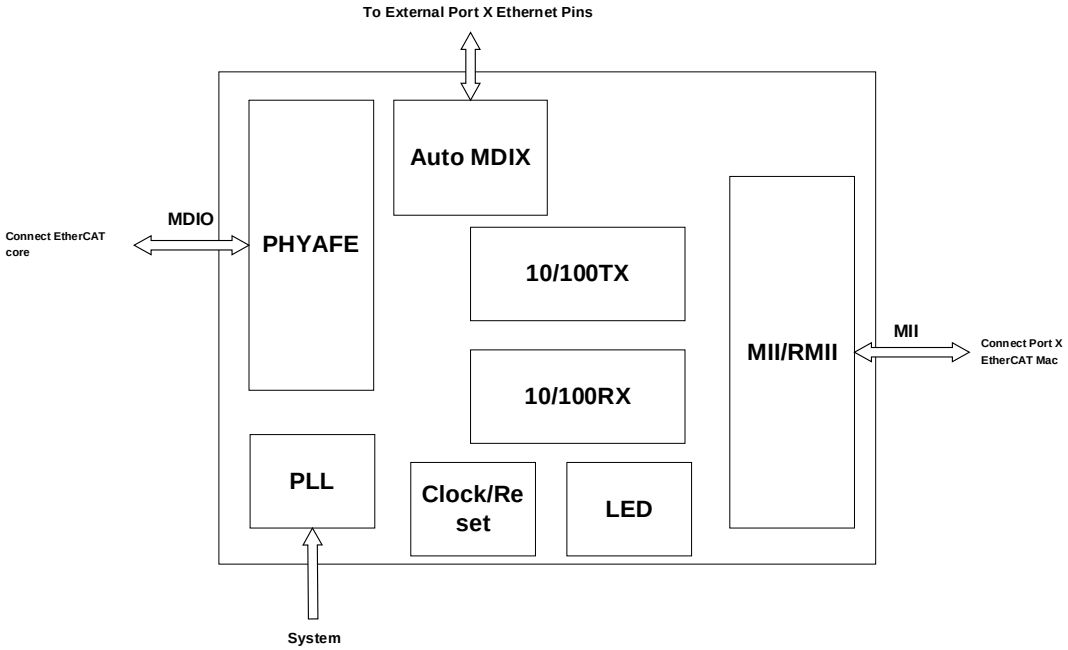
### 50.8.1. Overview

ESC contains PHYs A and B, there are identical in functionality. The PHY A connects to the EtherCAT port 0 or 2. The PHY B connects to EtherCAT port 1. These PHYs interface with their respective MAC via an internal MII interface. The PHYs comply with the IEEE 802.3 Physical Layer for Twisted Pair Ethernet and can be configured for full duplex 100 Mbps (100BASE-TX) Ethernet operation. All PHYs registers follow the IEEE 802.3 specified MII management register set and are fully configurable.

### 50.8.2. Characteristics

- Fully IEEE 802.3 100 Base-TX compliant and supports EEE
- Auto negotiation and parallel detection capability for automatic speed and duplex selection
- Supports MII and RMII interfaces
- Auto polarity correction in 10Base-T
- Supports Auto-MDIX function for Plug-n-Play
- Programmable loopback mode for diagnostic
- Supports programmable LED output for different applications power on LED Self-Test
- Supports WOL(Wake-On-Lan) functionality

**Figure 50-33.PHY functional block diagram**




## 50.8.3. Functional Overview

### Operation Mode

100BASE-TX: In the transmitter, the data stream from the MAC interfaces is 4B/5B encoded, serialized, scrambled, and coded with a MLT3 encoder. In the receiver, the data stream from the medium is recovered, decoded from MLT3, descrambled, parallelized, and 5B/4B decoded into 4-bit data.

When there's no data to be transmitted, the system informs its link partner with Low Power Idle (LPI) signaling and then enter transmitter power-saving mode. On the other hand, when receiving the LPI signal from the link partner, the system enters receiver power-saving mode. Only periodical signaling is used to keep the link alive.

### MII Interface

The Media Independent Interface (MII) is an interface, defined in IEEE 802.3u, between the MAC and PHY. The clock rate is equal to 2.5MHz for 10Mbps transmission and 25MHz for 100Mbps transmission.The MAC transmits and receivers data synchronously with TXCLK and RXCLK which are generated by PHY.

TXEN is asserted, TXD[3:0] is accepted for transmission by the PHY. Assertion of TXER while TXEN is asserted indicates transmit coding error. The combination of TXEN de-asserted, TXER asserted, and TXD[3:0] equal to 0001 shows a request to enter (or remain) in low power state. TXEN, TXER and TXD[3:0] are synchronously sampled with TXCLK.

When RXDV is asserted, RXD[3:0] transfer the recovered data from the PHY to MAC.

Assertion of RXER indicates a receive error. The combination of RXDV de-asserted, RXER asserted, and RXD[3:0] equal to 0001 informs it's LPI client (say MAC) that the link partner is in the low power state. CRS is asserted when the PHY is transmitting or receiving. COL is asserted when the PHY detects a collision. RXDV, RXER and RXD are synchronous with RXCLK.

### SMI Interface

The Serial Management Interface (SMI) can be used to transfer control and status information between the Station Management (STA) and the PHY. Users can also access the internal register settings of the PHY with SMI. The MDIO is a bidirectional signal, mainly composed of command (r/w) field and data field, and synchronous with MDC. The MDIO pin should be pulled-up when there's no driving signal.

### Automatic MDI/MDIX and Polarity Configuration

Automatic MDI/MDIX configuration is intended to eliminate the need for external crossover cables between two devices. PHY can do MDI/MDIX configuration automatically so that transmission and reception work normally. The MDI/MDIX configuration can also be determined by setting the register manually.

PHY can correct the polarity errors on the pairs of cable automatically.

### Loopback Modes

The loopback mode provides a diagnostic function to perform the transmission and reception, so it can tests the transmit and receive data paths.

### Wake-On-LAN

Wake-On-LAN is implemented using a special network message called a magic packet. The magic packet contains the MAC address of the destination device. The listening device waits for a legal magic packet addressed to it and then activates system wake-up procedure.

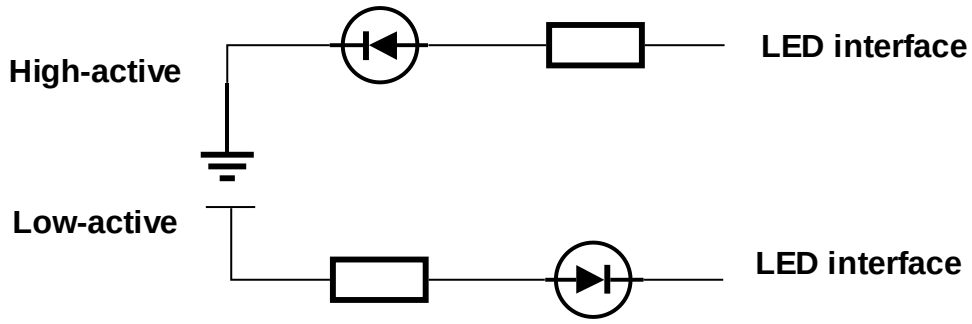
When Wake-on-Lan function is enabled, the PHY will send a interrupt after a legal magic packet is received.

### LED Modes

There are 3 LED interface used to control LED status for link status, speed and duplex mode indicating.

There are two LED connection type: high-active and low-active, showed ***Figure 50-34. LED connect diagram*** below:

**Figure 50-34. LED connect diagram**




Once led interface is connected as high-active type, the interface PHY_LED_POL should be tied to zero.

Once led interface is connected as low-active type, the interface PHY_LED_POL should be tied to one.

The LED status information is defined as below:

- Link LED:
  - on: Link is up
  - off: Link is down
  - flush: Data transmission
- Speed LED:
  - on: 100M
  - off: 10M
  - flush: N/A
- Duplex LED:
  - on: Full duplex
  - off: Half duplex
  - flush: Collision

The flush period of LED is 33 milliseconds.

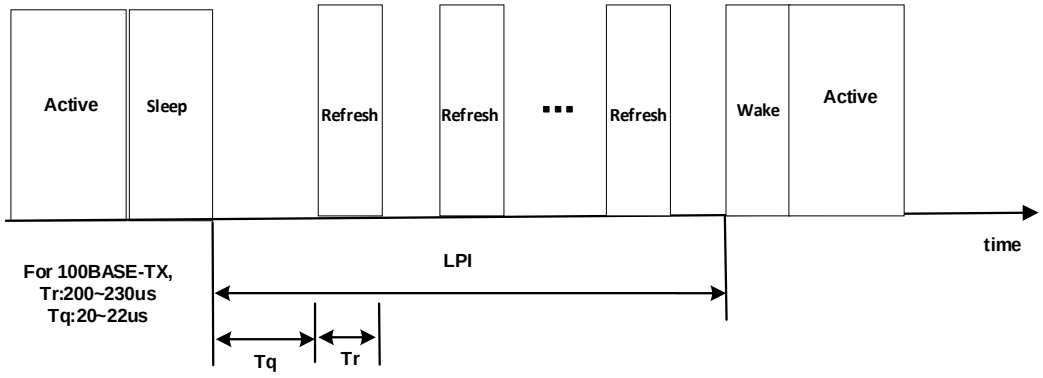
### LPI Signaling

When the LPI client issues a LPI request, the PHY transmits Sleep symbols to inform its link partner that the local PHY is going to enter the LPI state. The PHY enters LPI state after transmitting Sleep symbols. During the LPI state, only Refresh symbols is transmitted periodically. When the LPI client requests to leave the LPI state, the PHY transmits Wake symbols to ask the link partner to wake-up for further transmission.

When receiving the Sleep symbols from its link partner, the PHY knows that the remote PHY is going to enter the LPI state. After the remote PHY stops transmitting, the local PHY can turn off some circuits to save power. During the LPI state, the PHY uses Refresh symbols to update its filter coefficients and adjust timing. When receiving Wake symbols from its link

partner, the PHY goes back to normal operation from LPI state before a specified recovery time.

**Figure 50-35. 100Base-TX LPI**




## 50.8.4. PHY Register definition

### Page 0 Registers

#### PHY control Register (PHY_MII_CTL)

Address offset: 0x00

Reset value: 0x3100

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MR_MAIN_REST | LOOPBACK_EN | FORCE_SPEED(LSB) | MR_AUTONEG_EN | POWERDOWN | Reserved | MR_RESTART_AUTONEG | FORCE_DUPLEX | COL_TEST | FORCE_SPEED(MSB) | UNIDIRECTIONAL_EN | Reserved | | | | |
| rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | MR_MAIN_REST | Main Reset<br>Reset the status and control register of PHY to their default values and self-clearing<br>1: Reset<br>0: Normal |
| 14 | LOOPBACK_EN | Loopback Enable<br>When in loopback mode, the duplex will set to full duplex mode and Auto-Negotiation capability will be disabled automatically |
| 13 | FORCE_SPEED(LSB) | Force Speed LSB<br>Bit 13 and 6 combines the speed selection. It is only valid when MR_AUTONEG_EN = 0. When fiber mode is enable, internal speed will be set to 100M automatically and ignores this field setting.<br>00: 10M |

| | | |
|---|---|---|
| | | 01: 100M<br>1X: Reserved |
| 12 | MR_AUTONEG_EN | Auto Negotiation Enable<br>Result of this bit should be OR'ed with I_FXEN and EN_FX to determine internal final Auto-Negotiation enable signal (Auto negotiation will be disabled in Fiber mode). Please also note that when LOOPBACK_EN is set to 1, internal Auto-Negotiation capability will also be disabled automatically by the PHY and will not reflect to this bit.<br>1: Auto Negotiation Enable<br>0: Auto Negotiation Disable(Force 10M or 100M mode, according to the setting on FORCE_SPEED and FORCE_DUPLEX) |
| 11 | POWERDOWN | Power Down Mode<br>Active high to program the PHY into power down (power down analog TX analog RX, analog AD) |
| 10 | Reserved | Must be kept at reset value |
| 9 | MR_RESTART_AUTONEG | Re-Start Auto Negotiation<br>This bit will be self-cleared by the PHY after programming 1'b1 to this bit. Writing 1'b0 to this bit has no effect<br>1: Restart Auto-Negotiation<br>0: Normal |
| 8 | FORCE_DUPLEX | Force Duplex Mode<br>This bit is only valid when MR_AUTONEG_EN = 1'b0. Result of this bit should be OR'ed with LOOPBACK_EN to determine internal final duplex capability<br>1: Full Duplex (Default)<br>0: Half Duplex |
| 7 | COL_TEST | Collision Test<br>When this bit is assert, the PHY will assert COL signal within 512 BT in response to assertion of TX_EN and will de-assert the COL signal within 4 BT when connected to MII or 16 BT when connected to GMII in response to the de-assertion of TX_EN |
| 6 | FORCE_SPEED(MSB) | Force Speed MSB<br>Please refer to bit 13 |
| 5 | UNIDIRECTIONAL_EN | Unidirectional Enable<br>Enable the ability to encode and transmit data from the MII / GMII interface regardless of whether the PHY has determined that a valid link has been established. This ability is only valid when Auto-Negotiation is disabled and duplex mode is full |
| 4:0 | Reserved | Must be kept at reset value |

**PHY status Register (PHY_MII_STATUS)**

Address offset: 0x01

Reset value: 0x79C9

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 100BASE_T4 | 100BASE-X_FULL_DPX | 100BASE-X_HALF_DPX | 10BASE-T_FULL_DPX | 10BASE-T_HALF_DPX | 100BASE _T2_FULL_DPX | 100BASE_T2_HALF_DPX | EXTENDED_STATUS | UNIDIRECTIONAL_ABLT | MF_PRB_SUP | MR_AUTONEG_CPLT | REMOTE_FAULT | AUTONEG_ABLT | LINK_STATUS | JABBER_DETECT | EXTENDED_CAP |
| r | r | r | r | r | r | r | r | r | r | r | r | r | rc_r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | 100BASE_T4 | 100BASE T4 Capability<br>Not Supported. Will be 0 all the time |
| 14 | 100BASE-X_FULL_DPX | 100BASE TX Full Duplex Capable<br>1: PHY is 100BASE-X full duplex capable<br>0: PHY is not 100BASE-X full duplex capable |
| 13 | 100BASE-X_HALF_DPX | 100BASE TX Half Duplex Capable<br>1: PHY is 100BASE-X half duplex capable<br>0: PHY is not 100BASE-X half duplex capable |
| 12 | 10BASE-T_FULL_DPX | 10BASE-T Full Duplex Capable<br>1: PHY is 10BASE-T full duplex capable<br>0: PHY is not 10BASE-T full duplex capable |
| 11 | 10BASE-T_HALF_DPX | 10BASE-T Half Duplex Capable<br>1: PHY is 10BASE-T half duplex capable<br>0: PHY is not 10BASE-T half duplex capable |
| 10 | 100BASE _T2_FULL_DPX | 100BASE T2 Full Duplex Capable<br>Not Supported. Will be 0 all the time |
| 9 | 100BASE_T2_HALF_DPX | 100BASE T2 Half Duplex Capable<br>Not Supported. Will be 0 all the time |
| 8 | EXTENDED_STATUS | Extended Status<br>1: Extended status information in register 15<br>0: No extended status information in register 15 |
| 7 | UNIDIRECTIONAL_ABLT | Unidirectional Ability<br>1: PHY able to transmit from MII/GMII regardless of whether the PHY has determined that a valid link has been established<br>0: PHY able to transmit from MII/GMII only when the PHY has determined that a valid link has been established |
| 6 | MF_PRB_SUP | Preamble Suppression Capability<br>1: PHY can accept management frames with preamble suppression |

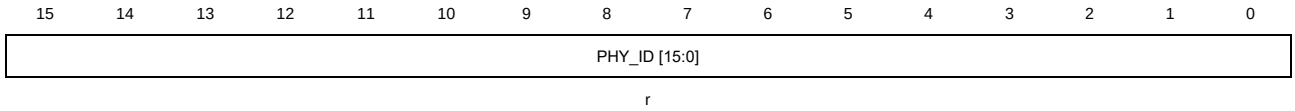
| | | |
|---|---|---|
| | | 0: PHY cannot accept management frames with preamble suppression |
| 5 | MR_AUTONEG_CPLT | Auto Negotiation Complete<br>1: Auto-negotiate completed<br>0: Auto-negotiate incomplete |
| 4 | REMOTE_FAULT | Remote Fault Detection<br>1: Remote Fault<br>0: Normal |
| 3 | AUTONEG_ABLT | Auto Negotiation Ability<br>1: PHY can Auto-Negotiate<br>0: PHY can not Auto-Negotiate |
| 2 | LINK_STATUS | Link Status<br>1: Link is up<br>0: Link is down<br>Note: When the PHY is programmed into loopback mode, ***PHY control Register (PHY_MII_CTL)*** BIT14 (LOOPBACK_EN is set to 1'b1), the PHY will be forced linkup. Link Status should updated according in this bit to make MAC work properly. This bit is implemented with a latching high function, such that the occurrence of a link failure condition will cause this bit to become cleared and remain cleared until it is read. |
| 1 | JABBER_DETECT | Jabber Condition Detected<br>1: Jabber RX and TX condition detected<br>0: Normal |
| 0 | EXTENDED_CAP | Extended Register Capability<br>1: Extended register capabilities<br>0: Basic register set capabilities only |

### PHY Identifier Register (PHY_ID_REG)

Address offset: 0x02

Reset value: 0x0044

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PHY_ID [15:0] | PHY ID bit [31-16]<br>OUI (bits 3-18). OUI =00-11-05 |

### PHY Version Register (PHY_VER_REG)

Address offset: 0x03

Reset value: 0x1400

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PHY_ID[5:0] | | | | | | PHY_MANF[5:0] | | | | | | PHY_REV_NO[3:0] | | | |
| r | | | | | | r | | | | | | r | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | PHY_ID[5:0] | PHY ID bit [5-0]<br>OUI bits 19-24 |
| 9:4 | PHY_MANF[5:0] | Manufacturer's Model Number<br>Manufacturer's Model Number (bits 5-0) where [5:4] = architecture version |
| 3:0 | PHY_REV_NO[3:0] | Revision Number (bits3-0) Register 3, bit 0 is LS bit of PHY Identifier |

### Auto-Negotiation Advertisement Register (PHY_AUTONEG_ADV)

Address offset: 0x04

Reset value: 0x0101

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_PAGE_EN | Reserved | REMOTE_FAULT_EN | EXTENED_NEXT_PAGE | ASYM_PAUSE | PAUSE | 100BASE_T4 | 100BASE_TX_FULL_DPX | 100BASE_TX_HALF_DPX | 10BASE-T_FULL_DPX | 10BASE-T_HALF_DPX | SLCT_FLD[4:0] | | | | |
| rw | | rw | rw | rw | rw | r | rw | rw | rw | rw | r | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | NEXT_PAGE_EN | Next Page Enable<br>1: Set to use Next Page<br>0: Not to use Next Page |
| 14 | Reserved | Must be kept at reset value |
| 13 | REMOTE_FAULT_EN | Remote Fault Detection Enable<br>1: Auto Negotiation Fault Detected<br>0: No Remote Fault |
| 12 | EXTENED_NEXT_PAGE | Extended Next Page<br>Not supported in the PHY. Should be wrote 0 all the time |
| 11 | ASYM_PAUSE | Asymmetric Pause Capability<br>Technology Ability A6<br>1: Asymmetric Pause capable<br>0: Asymmetric Pause non-capable |
| 10 | PAUSE | Pause Capability<br>Technology Ability A5<br>1: Pause capable |

0: Pause non-capable

| | | |
|---|---|---|
| 9 | 100BASE_T4 | 100BASE-T4 Capable<br>Not supported in the PHY. Should be wrote 0 all the time |
| 8 | 100BASETX_FULL_DPX | 100BASE-X Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY_MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8(FORCE_DUPLEX), when FORCE_SPEED is 2'b01 and FORCE_DUPLEX is 1'b1, then this bit will be 1'b1 and vice versa |
| 7 | 100BASETX_HALF_DPX | 100BASE-X Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY_MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8 (FORCE_DUPLEX). when FORCE_SPEED is 2'b01 and FORCE_DUPLEX is 1'b0, then this bit will be 1'b1 and vice versa |
| 6 | 10BASE-T_FULL_DPX | 10BASE-T Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY_MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8 (FORCE_DUPLEX). When FORCE_SPEED is 2'b00 and FORCE_DUPLEX is 1'b1, then this bit will be 1'b1 and vice versa |
| 5 | 10BASE-T_HALF_DPX | 10BASE-T Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY_MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8 (FORCE_DUPLEX). when FORCE_SPEED is 2'b00 and FORCE_DUPLEX is 1'b0, then this bit will be 1'b1 and vice versa |
| 4:0 | SLCT_FLD[4:0] | Identifies Type of Message<br>Forced to 5'h01 all the time |

### Auto-Negotiation Link Partner(LP) Ability Register (PHY_LP_ABILITY)

Address offset: 0x05

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_PAGE | ACKNOWLEDGE | REMOTE_FAULT | EXTENED_NEXT_P | ASYM_PAUSE | PAUSE | 100BASET4 | 100BASETX_FULL_ | 100BASETX_HALF | 10BASE-T_FULL_D | 10BASE-T_HALF_ | SELECTOR_FIELD[4:0] | | | | |

| | | | AGE | | | | DPX | _DPX | PX | DPX | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | NEXT_PAGE | Link Partner Next Page Request<br>1: Link Partner is requesting Next Page function<br>0: Base Page is requested |
| 14 | ACKNOWLEDGE | Link Partner ACKNOWLEDGE Received<br>1: Link partner acknowledge Received Successfully<br>0: Not Received |
| 13 | REMOTE_FAULT | Link Partner Detects Remote Fault<br>1: Auto Negotiation Fault Detected<br>0: No Remote Fault |
| 12 | EXTENED_NEXT_PAGE | Extended Next Page |
| 11 | ASYM_PAUSE | Link Partner Asymmetric Pause Capable<br>Technology Ability A6<br>1: Asymmetric Pause capable<br>0: Asymmetric Pause non-capable |
| 10 | PAUSE | Link Partner Symmetric Pause Capable<br>Technology Ability A5<br>1: Symmetric Pause capable<br>0: Symmetric Pause non-capable |
| 9 | 100BASET4 | Technology Ability A4<br>Link Partner 100BASE-T4 Capable |
| 8 | 100BASETX_FULL_DPX | Link Partner 100BASE-X Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |
| 7 | 100BASETX_HALF_DPX | Link Partner 100BASE-X Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |
| 6 | 10BASE-T_FULL_DPX | Link Partner 10BASE-T Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |
| 5 | 10BASE-T_HALF_DPX | Link Partner 10BASE-T Half Duplex Capable |

| | | |
|---|---|---|
| | | 1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1'b1 all the time |
| 4:0 | SELECTOR_FIELD[4:0] | Link Partner Identifies Type of Message<br>Should be 5'h01 |

### Auto-Negotiation Expansion Register (PHY_AUTONEG_EXP)

Address offset: 0x06

Reset value: 0x0064

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | NXT_PAGE_LOC_ABLE | NXT_PAGE_STO_LOC | PARA_DET_FAULT | LNK_PTNER_NP_ABLE | MR_NP_ABLE | PAGE_RECEIVED | LNK_PTNER_AN_ABLE |
| | | | | | | | | | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:7 | Reserved | Must be kept at reset value |
| 6 | NXT_PAGE_LOC_ABLE | Received Next Page Location Able<br>1: Received Next Page storage location is specified by bit 5<br>0: Received Next Page storage location is not specified by bit 5 |
| 5 | NXT_PAGE_STO_LOC | Received Next Page Storage Location<br>1: Link Partner next page are stored in ***Auto-Negotiation Next Page Received Register (PHY_AUTONEG_NEXT_PAGE_RECEIVE)***<br>0: Link Partner next page are stored in ***Auto-Negotiation Link Partner(LP) Ability Register (PHY_LP_ABILITY)*** |
| 4 | PARA_DET_FAULT | Parallel Detect Fault<br>1: Local Device Parallel Detection Fault<br>0: No fault detected |
| 3 | LNK_PTNER_NP_ABLE | Link Partner Next Page Capable<br>1: Link Partner is Next Page Able<br>0: Link Partner is not Next Page Able |
| 2 | MR_NP_ABLE | Next Page Capable<br>1: Local device is Next Page Able<br>0: Local device is not Next Page Able |
| 1 | PAGE_RECEIVED | New Page Received<br>1: A New Page has been received<br>0: A New Page has not been received |

| | | |
|---|---|---|
| 0 | LNK_PTNER_AN_ABLE | Link Partner Auto Negotiation Able<br>1: Link Partner is Auto-negotiation able<br>0: Link Partner is not Auto-negotiation able |

**Auto-Negotiation Next Page Transmit Register (PHY_AUTONEG_NEXT_PAGE_TRANSMIT)**

Address offset: 0x07

Reset value: 0x2001

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_PAGE | Reserved | MESSAGE_PAGE | ACKNOWLEDGE2 | TOGGLE | MSG_UFMT_CODE_FIELD[10:0] | | | | | | | | | | |
| rw | | rw | rw | r | rw | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | NEXT_PAGE | The Last Next Page<br>Indicated whether this is the last next page<br>1: Additional next page will follow<br>0: This is the last next page |
| 14 | Reserved | Must be kept at reset value. |
| 13 | MESSAGE_PAGE | Message Page or Unformatted Page<br>Indicated this is the message page or unformatted page<br>1: Message Page<br>0: Unformatted Page |
| 12 | ACKNOWLEDGE2 | The Ability to Comply with the Message<br>1: Will comply with the message<br>0: Can not comply with the message |
| 11 | TOGGLE | Toggle<br>The toggle bit will calculated by hardware automatically, SW can ignore |
| 10:0 | MSG_UFMT_CODE_FIELD[10:0] | Message/Unformatted Code Field |

**Auto-Negotiation Next Page Received Register (PHY_AUTONEG_NEXT_PAGE_RECEIVE)**

Address offset: 0x08

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NP_RX[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | NP_RX[15:0] | Next Page Received from Link Partner |

### MMD Access Control Register (MMD_CTL)

Address offset: 0x0D

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FUN[1:0] | | Reserved | | | | | | | | | DEVAD[4:0] | | | | |
| rw | | | | | | | | | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:14 | FUN[1:0] | Function<br>00: address<br>01: data, no post increment<br>10: data, post increment on reads and writes<br>11: data, post increment on writes only |
| 13:5 | Reserved | Must be kept at reset value. |
| 4:0 | DEVAD[4:0] | Device Address |

### MMD Access Data Address Register (MMD_ADDR_DATA)

Address offset: 0x0E

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR_DATA[15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | ADDR_DATA[15:0] | Address Data<br>When bit 13.15:14==0, address register Otherwise, data register |

### PHY Extended Status Register (PHY_EXTENDED_STATUS)

Address offset: 0x0F

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1000BASE- | 1000BASE- | 1000BASE- | 1000BASE- | Reserved | | | | | | | | | | | |

| X_FULL_DPX | X_HALF_DPX | T_HALF_DPX | T_HALF_DPX | |
|---|---|---|---|---|
| r | r | r | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | 1000BASE-X_FULL_DPX | 1000BASE TX Full Duplex Capable<br>1: PHY is 1000BASE-X full duplex capable<br>0: PHY is not 1000BASE-X full duplex capable |
| 14 | 1000BASE-X_HALF_DPX | 1000BASE TX Full Duplex Capable<br>1: PHY is 1000BASE-X full duplex capable<br>0: PHY is not 1000BASE-X full duplex capable |
| 13 | 1000BASE-T_HALF_DPX | 1000BASE-T Full Duplex Capable<br>1: PHY is 1000BASE-T full duplex capable<br>0: PHY is not 1000BASE-T full duplex capable |
| 12 | 1000BASE-T_HALF_DPX | 1000BASE-T Half Duplex Capable<br>1: PHY is 1000BASE-T half duplex capable<br>0: PHY is not 1000BASE-T half duplex capable |
| 11:0 | Reserved | Must be kept at reset value. |

### Interrupt Status Register (INT_STS)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LNK_STS_CHG_INT | MGC_PKT_DET_INT | TX_LPI_RCV_INT | RX_LPI_RCV_INT | Reserved | | | | | | | | | | | |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | LNK_STS_CHG_INT | Link Status Change INT<br>0: Normal<br>1: Link status change |
| 14 | MGC_PKT_DET_INT | Magic Packet Detect INT<br>0: Normal<br>1: Magic packet detected |
| 13 | TX_LPI_RCV_INT | TX LPI Received INT<br>0: Normal<br>1: TX LPI received |
| 12 | RX_LPI_RCV_INT | RX LPI Received INT Mask |

0: Normal

1: RX LPI received

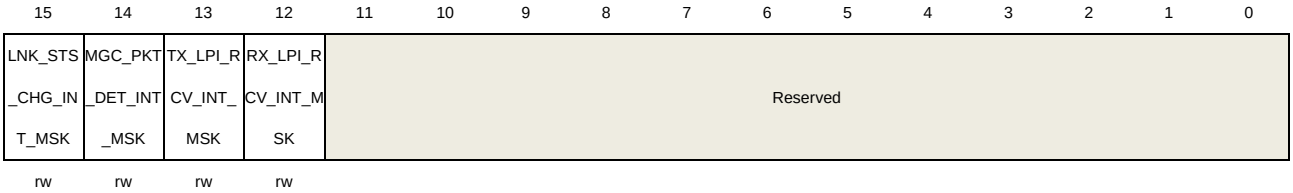
| | | |
|---|---|---|
| 11:0 | Reserved | Must be kept at reset value. |

### Interrupt Mask Register (INT_MASK)

Address offset: 0x11

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | LNK_STS_CHG_INT_MSK | Link Status Change INT Mask<br>When set to 1 and LINK_STS_CHG_INT bit in INT_STS register is assert to 1, the interface INT_SMI_INT_N/EXT_SMI_INT_N will be assert to 0(low active) to indicate that a magic packet detect interrupt is occured.<br>1: Normal<br>0: Interrupt is masked |
| 14 | MGC_PKT_DET_INT_MSK | Magic Packet Detect INT Mask<br>When set to 1 and MGC_PKT_DET_INT bit in INT_STS register is assert to 1, the interface INT_SMI_INT_N/EXT_SMI_INT_N will be assert to 0(low active) to indicate that a magic packet detect interrupt is occured.<br>1: Normal<br>0: Interrupt is masked |
| 13 | TX_LPI_RCV_INT_MSK | TX LPI Received INT Mask<br>When set to 1 and TX_LPI_RCV_INT bit in INT_STS register is assert to 1, the interface INT_SMI_INT_N/EXT_SMI_INT_N will be assert to 0(low active) to indicate that a TX LPI received interrupt is occured.<br>1: Normal<br>0: Interrupt is masked |
| 12 | RX_LPI_RCV_INT_MSK | RX LPI Received INT Mask<br>When set to 1 and RX_LPI_RCV_INT bit in INT_STS register is assert to 1, the interface INT_SMI_INT_N/EXT_SMI_INT_N will be assert to 0(low active) to indicate that a RX LPI received interrupt is occured.<br>1: Normal<br>0: Interrupt is masked |
| 11:0 | Reserved | Must be kept at reset value. |

### Loopback Control Register (PHY_LB_CTL)

Address offset: 0x12

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RG_LB_XMII2MAC | Reserved | RG_LB_PCS2MAC | RG_LB_PMA2MAC | RG_LB_AFE2MAC | RG_LB_EPG2EPC | Reserved | | RG_LB_MMII2PHY | Reserved | | | | | | |
| rw | rw | rw | rw | rw | rw | | | rw | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | RG_LB_XMII2MAC | XMII2MAC Loopback Enable |
| 14 | Reserved | Must be kept at reset value. |
| 13 | RG_LB_PCS2MAC | PCS2MAC Loopback Enable |
| 12 | RG_LB_PMA2MAC | PMA2MAC Loopback Enable |
| 11 | RG_LB_AFE2MAC | AFE2MAC Loopback Enable |
| 10 | RG_LB_EPG2EPC | EPG2EPC Loopback Enable |
| 9:8 | Reserved | Must be kept at reset value. |
| 7 | RG_LB_MMII2PHY | RMII2PHY Loopback Enable<br>Only valid when internal EPHY used. |
| 6:0 | Reserved | Must be kept at reset value. |

### PHY Global Configuration Register (PHY_GLOBAL_CONFIG)

Address offset: 0x13

Reset value: 0x0102

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | XMII_TXC_INV | XMII_RXC_INV | EN_FX | EN_WOL | FORCE_LINK | RG_WOL_RCV_BC | RG_WOL_CHK_PSWD | Reserved | | | | | MDI_MD[1:0] | |
| | | rw | rw | rw | rw | rw | rw | rw | | | | | | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:14 | Reserved | Must be kept at reset value. |
| 13 | XMII_TXC_INV | XMII TXCLK Inversed<br>0: No inverse about TXCLK on the XMII interface<br>1: Inverse TXCLK on the XMII interface |
| 12 | XMII_RXC_INV | XMII RXCLK Inversed |

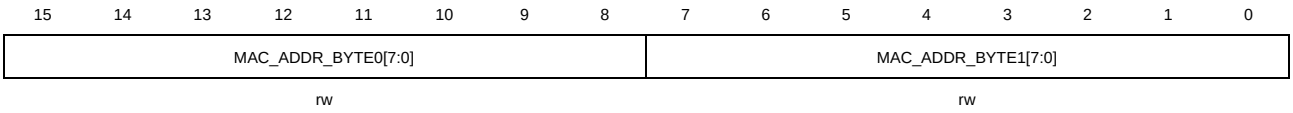
| | | |
|---|---|---|
| | | 0: No inverse about RXCLK on the XMII interface<br>1: Inverse RXCLK on the XMII interface |
| 11 | EN_FX | Fiber Enable<br>This bit will be OR'ed with EPHY_FXEN to determine the medium type of EPHY. Below shows the definition for OR'ed result<br>1: Fiber Mode<br>0: Twisted Pair Mode<br>Note: When EPHY is programmed into Fiber mode, Auto MDIX should be disabled automatically and force to MDI mode |
| 10 | EN_WOL | Wake-On-Lan Enable<br>Enable Magic Packet Detect Function |
| 9 | FORCE_LINK | Force Link Up<br>1: Force both 10/100M Module link up<br>0: Force Link or not depends upon FORCE_LINK_10 and/or FORCE_LINK_100 in 10/100M Configuration Register, respectively<br>Note: FORCE LINK works only when PHY is programmed into Force 10 or 100M mode. When PHY is programmed into auto negotiation mode, program 1'b1 to this bit has no effect (If this bit is programmed to 1'b1 ahead of auto negotiation is enabled, it should be disabled automatically by H/W |
| 8 | RG_WOL_RCV_BC | Enable Receive Broadcast Packet in WOL Mode<br>1: Enable receive broadcast magic packet<br>0: Otherwise |
| 7 | RG_WOL_CHK_PSWD | Enable SecureOn Password Check in WOL Mode<br>1: Enable SecureOn password check<br>0: Otherwise |
| 6:2 | Reserved | Must be kept at reset value. |
| 1:0 | MDI_MD[1:0] | MDI/MDIX Mode<br>Values on DUPCOLLED and RXER will be latched during power on reset and stored in these two bits<br>00: Force MDI Mode<br>01: Force MDIX Mode<br>10: Auto MDI/MDIX Detection (Default)<br>11: Reserved<br>MDI/MDIX mode will be set to 'Force MDI Mode' automatically when PHY is in either Loopback or Fiber mode |

**MAC Address Register 0 (RG_MAC_AADR_0)**

Address offset: 0x16

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



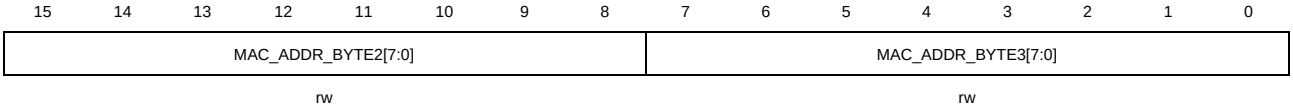

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | MAC_ADDR_BYTE0[7:0] | MAC Address Byte 0 in Transmission Order |
| 7:0 | MAC_ADDR_BYTE1[7:0] | MAC Address Byte 1 in Transmission Order |

### MAC Address Register 1 (RG_MAC_AADR_1)

Address offset: 0x17

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



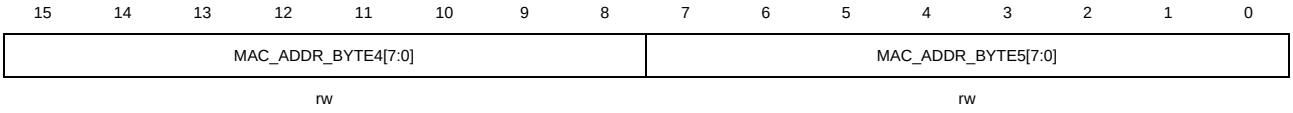

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | MAC_ADDR_BYTE2[7:0] | MAC Address Byte 2 in Transmission Order |
| 7:0 | MAC_ADDR_BYTE3[7:0] | MAC Address Byte 3 in Transmission Order |

### MAC Address Register 2 (RG_MAC_AADR_2)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | MAC_ADDR_BYTE4[7:0] | MAC Address Byte 4 in Transmission Order |
| 7:0 | MAC_ADDR_BYTE5[7:0] | MAC Address Byte 5 in Transmission Order |

### PHY Status Register (PHY_STATUS)

Address offset: 0x19

Reset value: 0x8800

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| LINE_NOSIG | Reserved | MR_AUTONEG_CPLT | A_TXSAVE80 | A_TXSAVE50 | JABTX_STATUS | MDIX_STATUS | mac_addr_byte5 | POLARITY_REV | FINAL_PAUSE_OUT | FINAL_PAUSE_IN | FINAL_DUPLEX | FINAL_SPEED[1:0] | FINAL_LINK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| r | | r | r | r | r | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | LINE_NOSIG | No Signal on the Medium<br>0: Signal detected in the medium<br>1: No signal detected in the medium |
| 14:13 | Reserved | Must be kept at reset value. |
| 12 | MR_AUTONEG_CPLT | Auto Negotiation Complete<br>1: Complete<br>0: Not Complete |
| 11 | A_TXSAVE80 | 10M Transmit 80% Amplitude Status<br>1: 80% TX Amplitude<br>0: Normal Amplitude |
| 10 | A_TXSAVE50 | 10M Transmit 50% Amplitude Status<br>1: 50% TX Amplitude<br>0: Normal Amplitude |
| 9 | JABRX_STATUS | Real Time RX Jabber Status<br>1: RX Jabber<br>0: No RX Jabber |
| 8 | JABTX_STATUS | Real Time TX Jabber Status<br>1: TX Jabber<br>0: No TX Jabber |
| 7 | MDIX_STATUS | MDIX STATUS<br>1: MDIX<br>0: MDI |
| 6 | POLARITY_REV | Polarity Status<br>1: Reversed<br>0: Normal |
| 5 | FINAL_PAUSE_OUT | Pause Out Capability<br>When auto negotiation is enabled, this bit is determined by ***Auto-Negotiation Advertisement Register (PHY_AUTONEG_ADV)*** BIT [11:10] and ***Auto-Negotiation Link Partner(LP) Ability Register (PHY_LP_ABILITY)*** BIT[11:10] after link up. When auto negotiation is disabled, this bit will be set to 1'b0 all the time<br>1: With Pause Out capability<br>0: Without Pause Out capability |
| 4 | FINAL_PAUSE_IN | Pause In Capability |

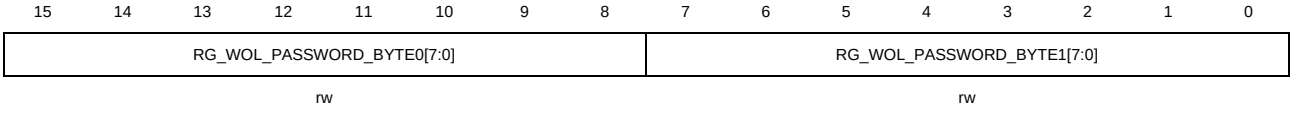
| | | |
|---|---|---|
| | | When auto negotiation is enabled, this bit is determined by ***Auto-Negotiation Advertisement Register (PHY AUTONEG ADV)*** BIT [11:10] and ***Auto-Negotiation Link Partner(LP) Ability Register (PHY_LP_ABILITY)*** BIT[11:10] after link up. When auto negotiation is disabled, this bit will be set to 1'b0 all the time<br>1: With Pause In capability<br>0: Without Pause In capability |
| 3 | FINAL_DUPLEX | Duplex Status<br>Before link up, S/W should ignore the status of this due to meaningless<br>1: Full Duplex<br>0: Half Duplex |
| 2:1 | FINAL_SPEED[1:0] | Speed Status<br>Before link up, S/W should ignore the status of this due to meaningless<br>1: 100M<br>0: 10M |
| 0 | FINAL_LINK | Link Status<br>1: Link Up<br>0: Link Down |

### Wake-On-Lan SecureOn Password Register 0 (RG_WOL_PASSWORD_0)

Address offset: 0x1A

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



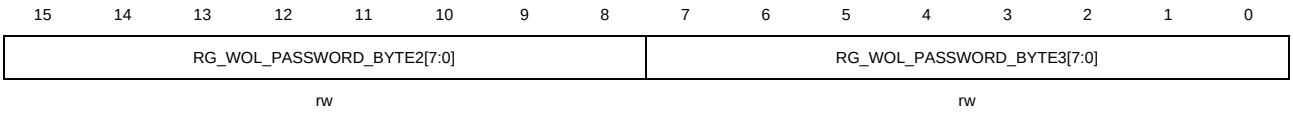

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | RG_WOL_PASSWORD_BYTE0[7:0] | SecureON Password Byte 0 in Transmission Order |
| 7:0 | RG_WOL_PASSWORD_BYTE1[7:0] | SecureON Password Byte 1 in Transmission Order |

### Wake-On-Lan SecureOn Password Register 1 (RG_WOL_PASSWORD_1)

Address offset: 0x1B

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|

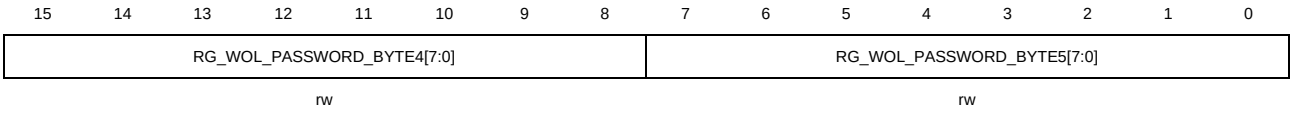
| | | |
|---|---|---|
| 15:8 | RG_WOL_PASSWORD_BYTE2[7:0] | SecureON Password Byte 2 in Transmission Order |
| 7:0 | RG_WOL_PASSWORD_BYTE3[7:0] | SecureON Password Byte 3 in Transmission Order |

### Wake-On-Lan SecureOn Password Register 2 (RG_WOL_PASSWORD_2)

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



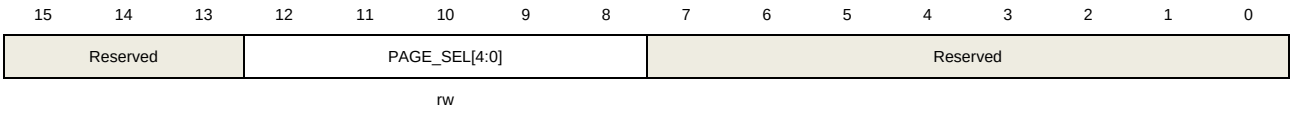

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | RG_WOL_PASSWORD_BYTE4[7:0] | SecureON Password Byte 4 in Transmission Order |
| 7:0 | RG_WOL_PASSWORD_BYTE5[7:0] | SecureON Password Byte 5 in Transmission Order |

### Page Selection Register (PHY_PAGE_SEL)

Address offset: 0x1F

Reset value: 0x003D

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:13 | Reserved | Must be kept at reset value. |
| 12:8 | PAGE_SEL[4:0] | MII Register Page Selection |
| 7:0 | Reserved | Must be kept at reset value. |

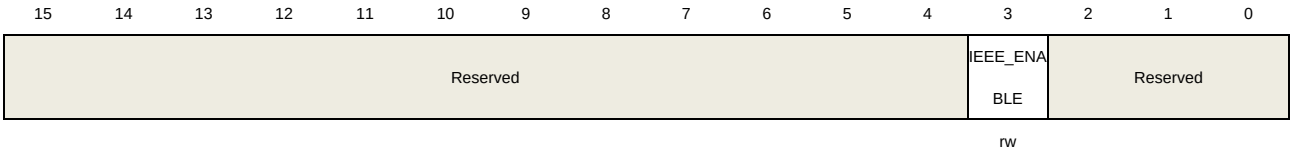
## Page 1 Registers

### EEE Configure Register (EEE_CFG)

Address offset: 0x17

Reset value: 0x0033

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| Bits | Fields | Descriptions |
|---|---|---|
| 15:4 | Reserved | Must be kept at reset value. |
| 3 | IEEE_ENABLE | Intelligent EEE Enable<br>Enable EPHY TX enter LPI state automatically when no data transmission |
| 2:0 | Reserved | Must be kept at reset value. |

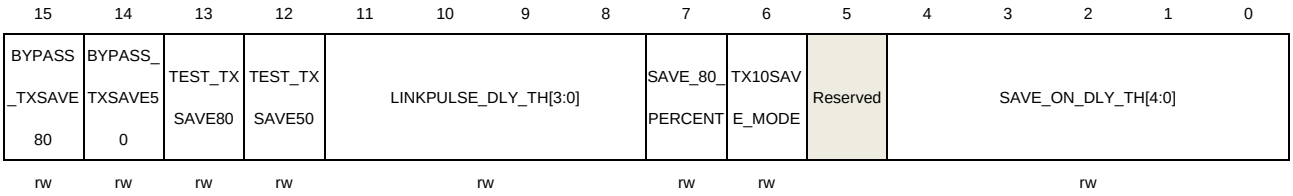
## Page 2 Registers

### 10M Power Save Control Register (PHY_10M_PWRSAVE)

Address offset: 0x17

Reset value: 0x04C8

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | BYPASS_TXSAVE80 | Bypass Normal Power Saving 80 Percent Path |
| 14 | BYPASS_TXSAVE50 | Bypass Normal Power Saving 50 Percent Path |
| 13 | TEST_TXSAVE80 | Power saving 80 percent Test Input<br>Valid only when BYPASS_TXSAVE80 is enabled |
| 12 | TEST_TXSAVE50 | Power saving 50 percent Test Input<br>Valid only when BYPASS_TXSAVE50 is enabled |
| 11:8 | LINKPULSE_DLY_TH[3:0] | Link Pulse Delay Generate Threshold<br>The threshold to delay transmission of TX link pulse. The time delayed is the value programmed x 2 cycles, where 1 cycle is equal to 40ns.<br>The default value set here is to delay 320ns |
| 7 | SAVE_80_PERCENT | Power Saving 80 Percent<br>1: Enable<br>0: Disable |
| 6 | TX10SAVE_MODE | 10M Power Saving Mode<br>1: 10M Power Saving mode enable<br>0: Normal mode without power saving |
| 5 | Reserved | Must be kept at reset value. |
| 4:0 | SAVE_ON_DLY_TH[4:0] | Power Saving on Delay Threshold |

### Analog Transmit Data Test and Control Register (PHY_TXDATA_CTRL)

Address offset: 0x18

Reset value: 0x1000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | FIR_10_SEL[2:0] | | | Reserved | | BYPASS_TX_10_DATA | BYPASS_TX_10_DATA | TEST_TX_100_DATA[1:0] | | Reserved | TEST_TX_10_DATA [4:0] | | | | |
| | rw | | | | | rw | rw | rw | | | rw | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14:12 | FIR_10_SEL[2:0] | 10M TX Filter Selection |
| 11:10 | Reserved | Must be kept at reset value. |
| 9 | BYPASS_TX_100_DATA | Bypass 100M Transmit data<br>1: bypass normal 100M data to analog and force TEST_TX_100_DATA to analog<br>0: Normal 100M data sent to analog block |
| 8 | BYPASS_TX_10_DATA | Bypass 10M Transmit data<br>1: bypass normal 10M data to analog and force TEST_TX_10_DATA to analog<br>0: Normal 10M data sent to analog block |
| 7:6 | TEST_TX_100_DATA[1:0] | 100M Test Data<br>Fed into analog block when BYPASS_TX_100_DATA is set to 1 |
| 5 | Reserved | Must be kept at reset value. |
| 4:0 | TEST_TX_10_DATA[4:0] | 10M Test Data<br>Fed into analog block when BYPASS_TX_10_DATA is set to 1 |

## Page 3 Registers

### DSPSM Control Register (PHY_DSPSM_CTRL)

Address offset: 0x11

Reset value: 0x8510

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | BYPASS_AGCSELD | Reserved | INI_100_AGCSEL[2:0] | | | Reserved | | | | BYPASS_BOOSTEND | TEST_BOOSTEND | Reserved | |
| | | | rw | | rw | | | | | | | rw | rw | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:13 | Reserved | Must be kept at reset value. |
| 12 | BYPASS_AGCSELD | Bypass AGCSEL<br>1: Bypass normal agcselD<br>0: Normal |
| 11 | Reserved | Must be kept at reset value. |
| 10:8 | INI_100_AGCSEL[2:0] | 100M Mode AGCSEL initial Value<br>Function of these 3 bits are different according to the setting of BYPASS_AGCSELD BYPASS AGCSELD = 1: Internal AGCSEL will fixed to INI_100_AGCSEL BYPASS AGCSELD = 0 : INI_100_AGCSEL will be loaded during DSPRST and internal boosten is 1'b0, and will be kept without change when internal boosten is low. Therefore, DSPSM can cover [boosten, AGCSEL] for the range of [0,000] and [1,000] to [1,111] |
| 7:4 | Reserved | Must be kept at reset value. |
| 3 | BYPASS_BOOSTEND | BoostenD BypassMode<br>Internal BoostenD will be skipped and TEST_BOOSTEND will be used to control for debugging purpose<br>1: Bypass mode<br>0: Normal mode |
| 2 | TEST_BOOSTEND | BoostenD Test Input<br>Force boostenD to this value when BYPASS_BOOSTEND = 1'b1 |
| 1:0 | Reserved | Must be kept at reset value. |

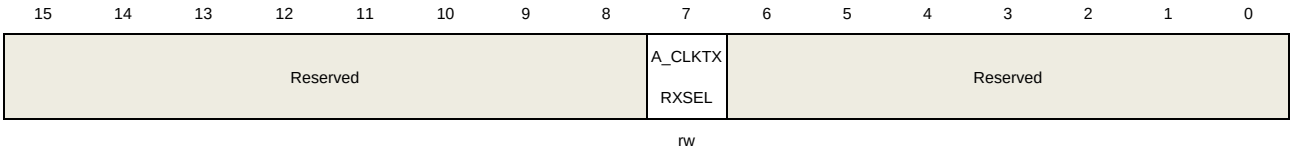
## Page 6 Registers

### Analog ADC Control Register (PHY_ADC_CTL)

Address offset: 0x10

Reset value: 0x5563

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | A_CLKTXRXSEL | Bypass RXCLK125 to TXCLK125<br>1: Internal RXCLK125 selects TXCLK125 |

0: Internal RXCLK125 selects RXCLK125 (Default)

| | | |
|---|---|---|
| 6:0 | Reserved | Must be kept at reset value. |

### Analog Pre-Gain and PLL Configuration Register (PHY_PGPLL_CTL)

Address offset: 0x12

Reset value: 0x0D00

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | SELCKADT | Reserved | | | | | | | | | | | | | |
| | rw | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14 | SELCKADT | Test Mode ADC Clock Select<br>This bit is only available when PHY is in AFE test mode,<br>other than AFE test mode, writing value to this bit has no effect<br>1: Select CKADTEST as ADC input clock<br>0: Select RXCLK125 as ADC input clock |
| 13:0 | Reserved | Must be kept at reset value. |

## Page 9 Registers

### Embedded Packet Generator and Checker Command Register (EPGC_CMD)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RG_EPG_EN | RG_EPG_DATA_TYPE[2:0] | | | RG_EPG_IPG_LEN_FIXED | RG_EPG_DASA_FIXED | RG_EPG_MODE[1:0] | | RG_EPG_LPI_INDICATION | RG_EPC_LOC | Reserved | | RG_EPC_CLR_CNT | RG_EPG_CLR_CNT | RG_EPG_PAUSE | RG_EPG_GO |
| rw | rw | rw | rw | rw | rw | rw | | rw | rw | | | rw | rw | rw | rw |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | RG_EPG_EN | Embedded Packet Generator Enable<br>This bit is used to enable embedded packet generator for debugging purpose. When it is asserted, internal tx data path will be switched to the embedded packet generator |
| 14:12 | RG_EPG_DATA_TYPE[2:0] | Embedded Packet Generator Mode<br>3'b000: all zeros; |

| | | |
|---|---|---|
| | | 3’b001: all ones;<br>3’b010: all 5s;<br>3’b011: all As;<br>3’b100: byte Increment;<br>3’b101: random;<br>3’b110: byte decrement |
| 11 | RG_EPG_IPG_LEN_FIXED | Inter-Packet-Gap Length Fixed<br>1: Inter-Packet-Gap length fixed and determined by RG_EPG_IPG_LEN<br>0: Inter-Packet-Gap length randomized by hardware |
| 10 | RG_EPG_DASA_FIXED | DA/SA Fixed<br>1: DA fixed to 00-01-02-03-04-05 and SA fixed to 0a-0b-0c-0d-0e-0f<br>0: DA/SA is configured by RG_EPG_DATA_TYPE |
| 9:8 | RG_EPG_MODE[1:0] | Embedded Packet Generator Mode<br>2’b00: single mode;<br>2’b01: burst mode;<br>2’b1x: continue Mode |
| 7 | RG_EPG_LPI_INDICATION | TX LPI indication<br>1: Transmit LPI<br>0: Normal |
| 6 | RG_EPC_LOC | EPC Location<br>1: EPC is located at TX path<br>0: EPC is located at RX path |
| 5:4 | Reserved | Must be kept at reset value. |
| 3 | RG_EPC_CLR_CNT | Embedded Packet Checker Counter Clear<br>High active to clear rx packet checker statistic counter, include total packet number counter and crc error counter, it is self-cleared |
| 2 | RG_EPG_CLR_CNT | Embedded Packet Generator Counter<br>Clear High active to clear tx packet generator statistic counter, include total packet generated counter, it is self-cleared |
| 1 | RG_EPG_PAUSE | Embedded Packet Generator Packet Generation Pause Combined with RG_EPG_GO to control packet generator, { RG_EPG_PAUSE, RG_EPG_GO } =<br>2’b01: start;<br>2’b11: pause;<br>2’b00: stop<br>Note: In single mode(RG_EPG_MODE[1:0]=2’b0), only the start command is valid. The other two commands will not work In burst mode(RG_EPG_MODE[1:0]=2’b01), all the three commands are valid, the pause command will pause packet generation, internal burst packet number counter will hold, and followed start command will continue the current burst |

generation. In continue mode(RG_EPG_MODE[1:0]=2'b10), all the three commands are valid, and the pause command behavior will be the same with the stop command.

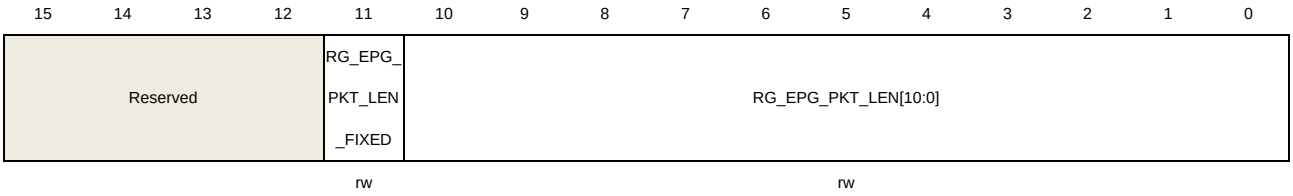
| | | |
|---|---|---|
| 0 | RG_EPG_GO | Embedded Packet Generator Packet Generation Go<br>Combined with RG_EPG_PAUSE to control packet generator, please refer to RG_EPG_PAUSE for the control command definition.<br>Note: When in single mode and continue mode, this bit will be self-cleared when generation task is finished. When in continue mode, only write zero to this bit can clear it. |

### Embedded Packet Generator Packet Length (EPG_PKT_LEN)

Address offset: 0x11

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



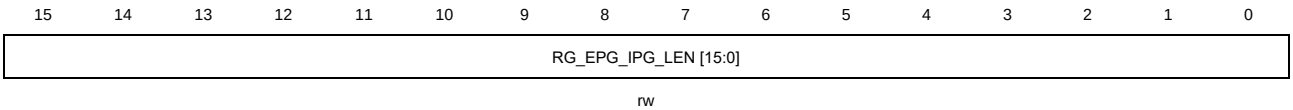

| Bits | Fields | Descriptions |
|---|---|---|
| 15:12 | Reserved | Must be kept at reset value. |
| 11 | RG_EPG_PKT_LEN_FIXED | Packet Length Fixed Enable<br>1: Packet length fixed and determined by rg_epg_pkt_len<br>0: Packet length randomized by hardware |
| 10:0 | RG_EPG_PKT_LEN[10:0] | Packet Length<br>Packet total length, include the DA/SA, data and FCS. It is only valid when rg_epg_pkt_len_fixed is 1'b1, counted by byte, otherwise, packet length will be randomized by hardware |

### Embedded Packet Generator Inter-Packet-Gap (epg_ipg_cfg)

Address offset: 0x12

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RG_EPG_IPG_LEN [15:0] | Inter-Packet-Gap Length |

**Embedded Packet Generator Burst Number High Data (EPG_BURST_PKT_NUM_CFG_HIGH)**

Address offset: 0x13

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RG_EPG_BURST_PKT_NUMBER_HIGH [15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RG_EPG_BURST_PKT_NUMBER_HIGH [15:0] | Packet Number of a Burst High Data Only valid in burst mode (RG_EPG_MODE==2'b10) |

**Embedded Packet Generator Burst Number Low Data (EPG_BURST_PKT_NUM_CFG_LOW)**

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RG_EPG_BURST_PKT_NUMBER_LOW [15:0] | | | | | | | | | | | | | | | |
| rw | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RG_EPG_BURST_PKT_NUMBER_LOW [15:0] | Packet Number of a Burst Low Data Only valid in burst mode (RG_EPG_MODE==2'b10) |

**TX Byte Counter High Data (TX_BYTE_CNT_HIGH)**

Address offset: 0x15

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TX_BYTE_CNT_HIGH[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | TX_BYTE_CNT_HIGH[15:0] | Embedded Packet Generator Packet Generation Byte Counter High Data(bit 31:16)<br>This counter will be cleared by asserting RG_EPG_CLR_CNT |

**TX Byte Counter Low Data (TX_BYTE_CNT_LOW)**

Address offset: 0x16

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TX_BYTE_CNT_LOW[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | TX_BYTE_CNT_LOW[15:0] | Embedded Packet Generator Packet Generation Byte Counter High Data(bit 15:0)<br>This counter will be cleared by asserting RG_EPG_CLR_CNT |

### TX Total Packet Counter High Data (TX_PKT_CNT_HIGH)

Address offset: 0x17

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TX_PKT_CNT_HIGH[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | TX_PKT_CNT_HIGH[15:0] | Embedded Packet Generator Total Packet Generation Counter High Data(bit 31:16)<br>This counter will be cleared by asserting RG_EPG_CLR_CNT |

### TX Total Packet Counter Low Data (TX_PKT_CNT_LOW)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TX_PKT_CNT_LOW[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

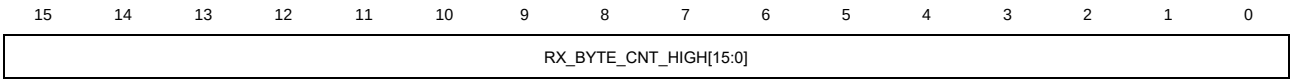
| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | TX_PKT_CNT_LOW[15:0] | Embedded Packet Generator Total Packet Generation Counter Low Data(bit 15:0)<br>This counter will be cleared by asserting RG_EPG_CLR_CNT |

### RX Byte Counter High Data (RX_BYTE_CNT_HIGH)

Address offset: 0x19

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



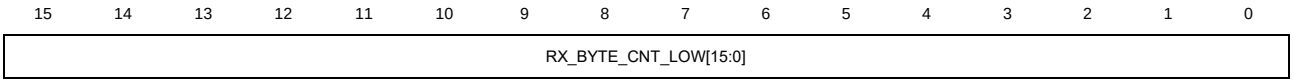

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RX_BYTE_CNT_HIGH[15:0] | Packet Received Byte Counter High Data(bit 31:16)<br>This counter will be cleared by asserting RG_EPC_CLR_CNT |

### RX Byte Packet Counter Low Data (RX_BYTE_CNT_LOW)

Address offset: 0x1A

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



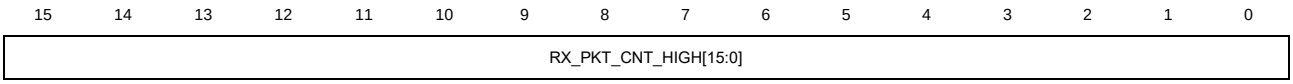

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RX_BYTE_CNT_LOW[15:0] | Packet Received Byte Counter Low Data(bit 15:0)<br>This counter will be cleared by asserting RG_EPC_CLR_CNT |

### RX Total Packet Counter High Data (RX_PKT_CNT_HIGH)

Address offset: 0x1B

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



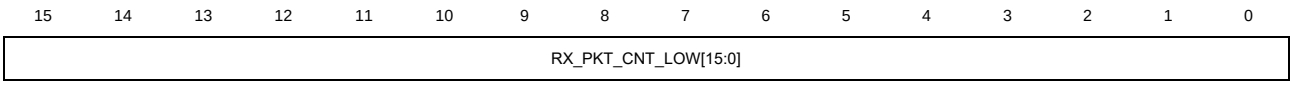

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RX_PKT_CNT_HIGH[15:0] | Total Packet Received Counter High Data(bit 31:16)<br>This counter will be cleared by asserting RG_EPC_CLR_CNT |

### RX Total Packet Counter Low Data (RX_PKT_CNT_LOW)

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



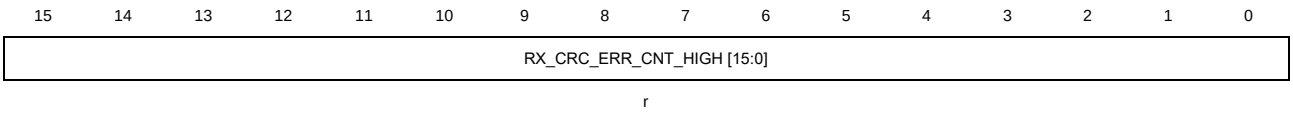
| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RX_PKT_CNT_LOW[15:0] | Total Packet Received Counter Low Data(bit 15:0)<br>This counter will be cleared by asserting RG_EPC_CLR_CNT |

### RX CRC Error Packet Counter High Data (RX_CRC_ERR_CNT_HIGH)

Address offset: 0x1D

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



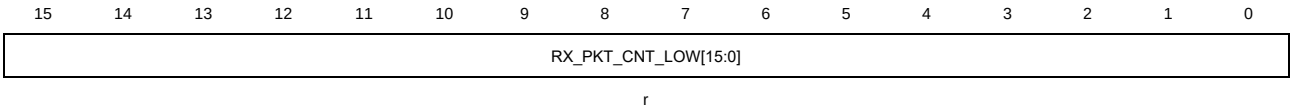

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RX_CRC_ERR_CNT_HIGH[15:0] | CRC Error Packet Received Counter High Data(bit 31:16)<br>This counter will be cleared by asserting RG_EPC_CLR_CNT |

### RX CRC Error Packet Counter Low Data (RX_CRC_ERR_CNT_LOW)

Address offset: 0x1E

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RX_CRC_ERR_CNT_LOW[15:0] | CRC Error Packet Received Counter Low Data(bit 15:0)<br>This counter will be cleared by asserting RG_EPC_CLR_CNT |

## MDIO Registers

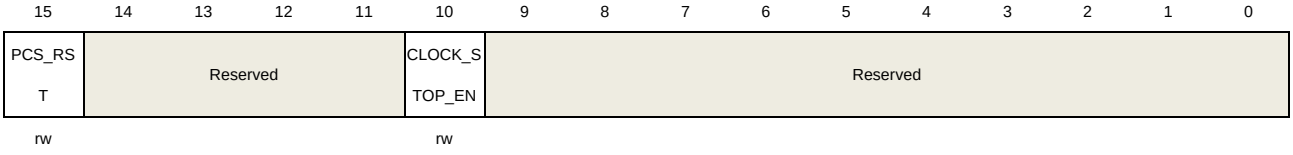
### PCS Control 1 Register (PCS_CTL_1)

Device Address: 0x3

Address offset: 0x00

Reset value: 0x0400

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| Bits | Fields | Descriptions |
|---|---|---|
| 15 | PCS_RST | PCS Reset<br>Reset the AN and PCS MMD registers to their default value and also cause a software reset in the PHY, it will be selfclearing after reset is finished |
| 14:11 | Reserved | Must be kept at reset value. |
| 10 | CLOCK_STOP_EN | xMII Receive Clock Stop Enable<br>Set to 1 to stop the receive xMII clock while it is signaling LPI otherwise it will keep the clock active |
| 9:0 | Reserved | Must be kept at reset value. |

### PCS Status 1 Register (PCS_STS_1)

Device Address: 0x3

Address offset: 0x01

Reset value: 0x0040

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | TX_LPI_REC | RX_LPI_REC | TX_LPI_INT | RX_LPI_INT | Reserved | CLOCK_STOP_CAP | Reserved | | | | | |
| | | | | r | r | r | r | | r | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:12 | Reserved | Must be kept at reset value. |
| 11 | TX_LPI_REC | TX PCS has Received LPI<br>It is implemented with a latching high function, such that the TX LPI signaling causes this bit to become one and remain one until it is read.<br>1: TX PCS has received LPI<br>0: LPI not received |
| 10 | RX_LPI_REC | RX PCS has Received LPI<br>It is implemented with a latching high function, such that the RX LPI signaling causes this bit to become one and remain one until it is read.<br>1: RX PCS has received LPI<br>0: LPI not received |
| 9 | TX_LPI_INT | TX PCS Receiving LPI Indication<br>1: TX PCS is currently receiving LPI<br>0: PCS is not currently receiving LPI |
| 8 | RX_LPI_INT | RX PCS Receiving LPI Indication<br>1: RX PCS is currently receiving LPI<br>0: PCS is not currently receiving LPI |

| | | |
|---|---|---|
| 7 | Reserved | Must be kept at reset value. |
| 6 | CLOCK_STOP_CAP | Transmit xMII Clock Stop Capable<br>1: RS may stop the transmit xMII clock during LPI<br>0: Transmit xMII clock not stoppable |
| 5:0 | Reserved | Must be kept at reset value. |

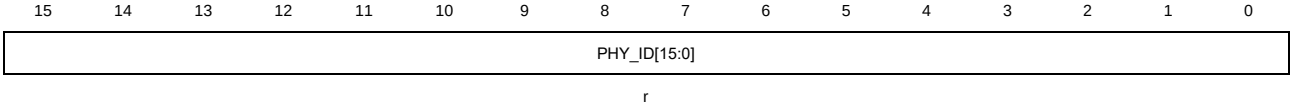
**PCS Device Identifier (PCS_ID)**

Device Address: 0x3

Address offset: 0x02

Reset value: 0x0044

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PHY_ID[15:0] | PHY ID bit[31-16]<br>OUI (bits 3-18) .OUI =00-11-05 |

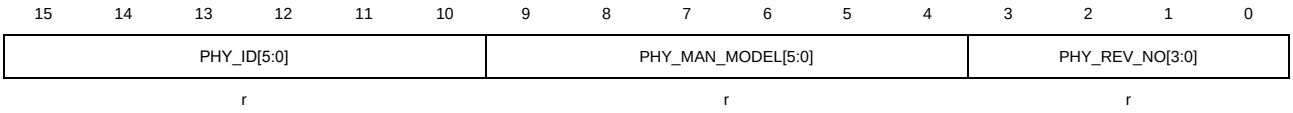
**PCS Device Version Register (PCS_VER)**

Device Address: 0x3

Address offset: 0x03

Reset value: 0x1400

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | PHY_ID[5:0] | PHY ID bit[15-10]<br>OUI bits 19-24 |
| 9:4 | PHY_MAN_MODEL[5:0] | Manufacturer's Model Number<br>Manufacturer's Model Number (bits 5-0) where [5:4] = architecture version |
| 3:0 | PHY_REV_NO[3:0] | Revision Number (bits3-0)<br>***PCS Device Identifier (PCS_ID)*** bit 0 is LS bit of PHY Identifier |

**PCS Package Register 0 (PCS_PKG_0)**

Device Address: 0x3

Address offset: 0x05

Reset value: 0x0089

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | AN_PST | TC_PST | DTE_XS_PST | PHY_XS_PST | PCS_PST | WIS_PST | PMD/PMA_PST | CLAUSS22_REG_PST |
| | | | | | | | | r | r | r | r | r | r | r | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | AN_PST | Auto-Negotiation Present In Package<br>Always 1 |
| 6 | TC_PST | TC Present In Package<br>Always 0 |
| 5 | DTE_XS_PST | DTE XS Present In Package<br>Always 0 |
| 4 | PHY_XS_PST | PHY XS Present In Package<br>Always 0 |
| 3 | PCS_PST | PCS Present In Package<br>Always 0 |
| 2 | WIS_PST | WIS Present In Package<br>Always 0 |
| 1 | PMD/PMA_PST | PMD/PMA Present In Package<br>Always 0 |
| 0 | CLAUSS22_REG_PST | Auto-Negotiation Present In Package<br>Always 1 |

**PCS Package Register 1 (PCS_PKG_1)**

Device Address: 0x3

Address offset: 0x06

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VENDER_SPEC_DEV2_PST | VENDER_SPEC_DEV1_PST | CL22_EXT_PST | Reserved | | | | | | | | | | | | |
| r | r | r | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | VENDER_SPEC_DEV2_PST | Vender Specific Device 2 Present in Package<br>Always 0 |
| 14 | VENDER_SPEC_DEV1_PST | Vender Specific Device 1 Present in Package<br>Always 0 |
| 13 | CL22_EXT_PST | Clause 22 Extension Present in Package<br>Always 0 |
| 12:0 | Reserved | Must be kept at reset value. |

### EEE Capability Register (EEE_CAP)

Device Address: 0x3

Address offset: 0x14

Reset value: 0x0002

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | 10GBASE_KR_EEE | 10GBASE_KX4_EEE | 1000BASE_KX_EEE | 10GBASE_T_EEE | 1000BASE_T_EEE | 100BASE_TX_EEE | Reserved |
| | | | | | | | | | r | r | r | r | r | rw | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:7 | Reserved | Must be kept at reset value. |
| 6 | 10GBASE_KR_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 5 | 10GBASE_KX4_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 4 | 1000BASE_KX_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 3 | 10GBASE_T_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 2 | 1000BASE_T_EEE | 1000Base-T EEE<br>1: EEE is supported for 1000BASE-T<br>0: EEE is not supported for 1000Base-T |
| 1 | 100BASE_TX_EEE | 100Base-TX EEE<br>1: EEE is supported for 100BASE-TX<br>0: EEE is not supported for 100BASE-TX |
| 0 | Reserved | Must be kept at reset value. |

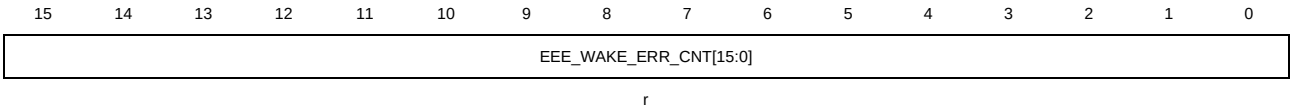
### EEE Wake Error Counter (EEE_WAKE_ERR_CNT)

Device Address: 0x3

Address offset: 0x16

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | EEE_WAKE_ERR_CNT[15:0] | EEE Wake Error Counter<br>Used to count wake time faults where the PHY fails to complete its normal wake sequence within the time required for the specific PHY type. It is cleared when it is read |

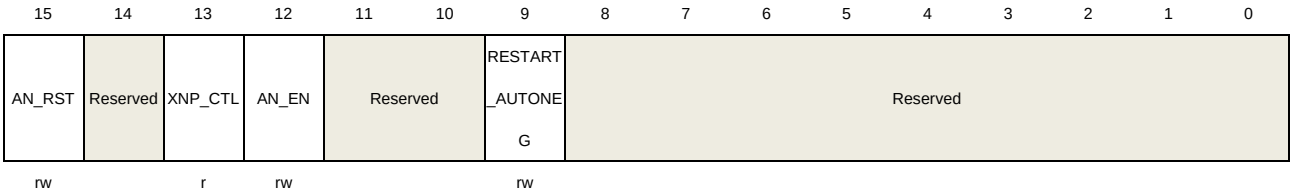
### AN Control Register (AN_CTL)

Device Address: 0x7

Address offset: 0x00

Reset value: 0x1000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | AN_RST | AN Reset<br>Reset the AN and PCS MMD registers to their default value and also cause a software reset in the PHY, it will be selfclearing after reset is finished |
| 14 | Reserved | Must be kept at reset value. |
| 13 | XNP_CTL | Extended Next Page Control<br>Not support and reserved 0 |
| 12 | AN_EN | Auto-Negotiation Enable<br>It is a copy of bit 12 in ***PHY control Register (PHY_MII_CTL)***<br>1: Auto-Negotiation Enable<br>0: Auto-Negotiation Disable |
| 11:10 | Reserved | Must be kept at reset value. |
| 9 | RESTART_AUTONEG | Restart Auto-Negotiation |

| | | |
|---|---|---|
| | | It is a copy of bit 9 in ***PHY control Register (PHY_MII_CTL)***. This bit is self-clearing<br>1: Restart Auto-Negotiation<br>0: Normal operation |
| 8:0 | Reserved | Must be kept at reset value. |

### AN Status Register (AN_STS)

Device Address: 0x7

Address offset: 0x01

Reset value: 0x0008

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | PAGE_RECEIVED | AN_COMPLETE | AN_ABILITY | AN_ABILITY | LINK_STATUS | Reserved | LP_AN_ABILITY |
| | | | | | | | | | r | r | r | r | r | | r |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:7 | Reserved | Must be kept at reset value. |
| 6 | PAGE_RECEIVED | Page Received<br>Set to 1 to indicate that a new link codeword has been received and stored in the an_lp_xnp_ability registers. The Page received bit will be reset to 0 on a read of the AN status register or the Auto-Negotiation expansion register. This bit is a copy of bit 1 in ***Auto-Negotiation Expansion Register (PHY_AUTONEG_EXP)***.<br>1: A new link codeword has been received<br>0: No link codeword has been received |
| 5 | AN_COMPLETE | Auto-Negotiation Complete<br>Set to one to indicate that the Auto-Negotiation has been completed, It will return a 0when Auto-Negotiation is disabled by clearing bit 12 in ***AN Control Register (AN_CTL)***. |
| 4 | AN_ABILITY | Remote Fault<br>Set to one to indicate that a remote fault condition has been detected. A remote fault is defined as Far-End-Fault when fiber mode is enable. It is implemented with a latching high function, such that the occurrence of a remote fault causes the bit 4 to become set and remain set until it is cleared. It will be reset to 0 on a read of the AN status register or the Status Register (It is a copy of bit 4 in ***PHY status Register (PHY_MII_STATUS)***). |
| 3 | AN_ABILITY | Auto-Negotiation Ability<br>Always set to one in twisted-pair mode to indicate that the PHY has the ability to perform Auto-Negotiation. It is a copy of bit 3 in ***PHY status*** |

| | | |
|---|---|---|
| | | ***Register (PHY_MII_STATUS)*** |
| 2 | LINK_STATUS | Link Status<br>Set to one to indicate that a valid link has been established. It will be implemented with a latching low function, such that the occurrence of a link fail condition causes the link status bit to become cleared and remain cleared until it is read. It is a copy of bit 2 in ***PHY status Register (PHY_MII_STATUS)*** |
| 1 | Reserved | Must be kept at reset value. |
| 0 | LP_AN_ABILITY | Link Partner Auto-Negotiation Ability<br>Set to one to indicate that the link partner is able to participate in the Auto-Negotiation function |

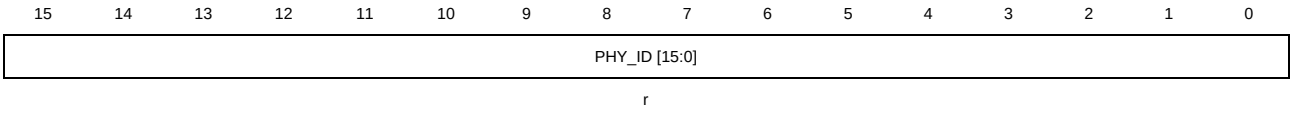
### Auto-Negotiation Device Identifier (AN_ID)

Device Address: 0x7

Address offset: 0x02

Reset value: 0x0044

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PHY_ID[15:0] | PHY ID bit[31-16]<br>OUI (bits 3-18). OUI =00-11-05 |

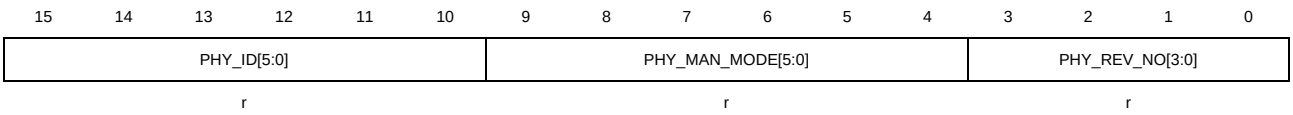
### Auto-Negotiation Device Version Register (AN_VER)

Device Address: 0x7

Address offset: 0x03

Reset value: 0x1400

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | PHY_ID[5:0] | PHY ID bit [15-10]<br>OUI bits 19-24 |
| 9:4 | PHY_MAN_MODE[5:0] | Manufacturer's Model Number<br>Manufacturer's Model Number (bits 5-0) where [5:4] = architecture version |

| | | |
|---|---|---|
| 3:0 | PHY_REV_NO[3:0] | Revision Number (bits3-0)<br>***PCS Device Identifier (PCS_ID)*** bit 0 is LS bit of PHY Identifier |

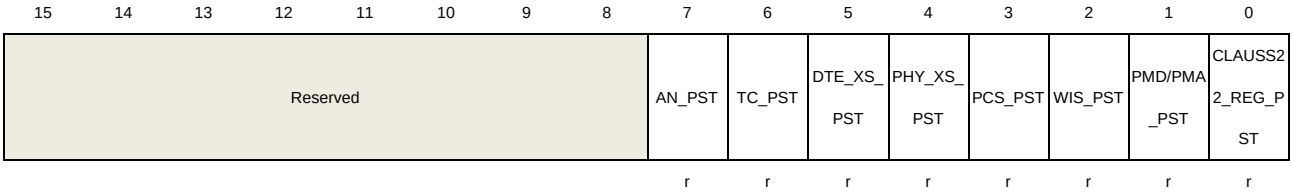
### AN Package Register 0 (AN_PKG_0)

Device Address: 0x7

Address offset: 0x05

Reset value: 0x0089

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | Reserved | Must be kept at reset value. |
| 7 | AN_PST | Auto-Negotiation Present In Package<br>Always 1 |
| 6 | TC_PST | TC Present In Package<br>Always 0 |
| 5 | DTE_XS_PST | DTE XS Present In Package<br>Always 0 |
| 4 | PHY_XS_PST | PHY XS Present In Package<br>Always 0 |
| 3 | PCS_PST | PCS Present In Package<br>Always 0 |
| 2 | WIS_PST | PMD/PMA Present In Package<br>Always 0 |
| 1 | PMD/PMA_PST | PMD/PMA Present In Package<br>Always 0 |
| 0 | CLAUSS22_REG_PST | Auto-Negotiation Present In Package<br>Always 1 |

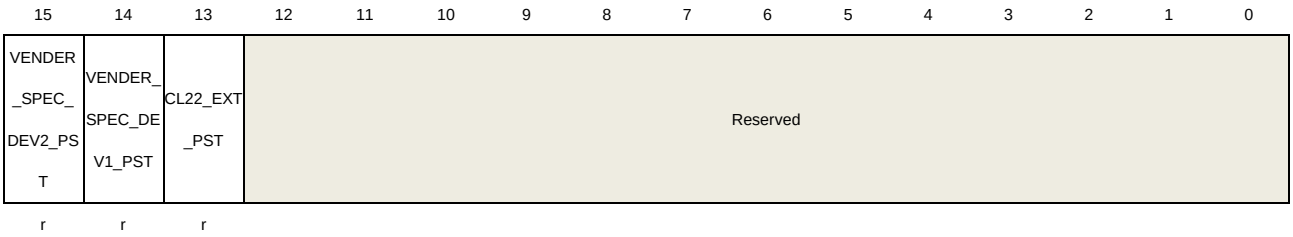
### AN Package Register 1 (AN_PKG_1)

Device Address: 0x7

Address offset: 0x06

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | VENDER_SPEC_DEV2_PST | Vender Specific Device 2 Present in Package<br>Always 0 |
| 14 | VENDER_SPEC_DEV1_PST | Vender Specific Device 1 Present in Package<br>Always 0 |
| 13 | CL22_EXT_PST | Clause 22 Extension Present in Package<br>Always 0 |
| 12:0 | Reserved | Must be kept at reset value. |

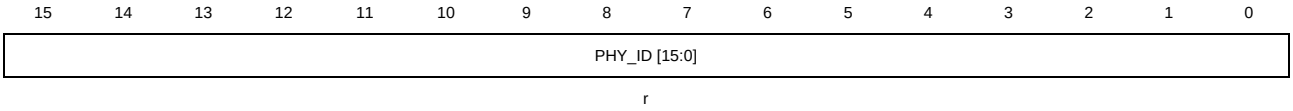
### Auto-Negotiation Device Identifier (AN_ID)

Device Address: 0x7

Address offset: 0x0E

Reset value: 0x0044

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PHY_ID[15:0] | PHY ID bit[31-16]<br>OUI (bits 3-18). OUI =00-11-05 |

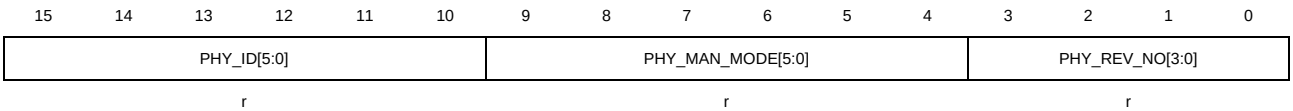
### Auto-Negotiation Device Version Register (AN_VER)

Device Address: 0x7

Address offset: 0x0F

Reset value: 0x1400

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:10 | PHY_ID[5:0] | PHY ID bit [15-10] |

| | | |
|---|---|---|
| | | OUI bits 19-24 |
| 9:4 | PHY_MAN_MODE[5:0] | Manufacturer's Model Number<br>Manufacturer's Model Number (bits 5-0) where [5:4] = architecture version |
| 3:0 | PHY_REV_NO[3:0] | Revision Number (bits3-0) Register 3, bit 0 is LS bit of PHY Identifier |

### Auto-Negotiation Advertisement Register (AN_ADV)

Device Address: 0x7

Address offset: 0x10

Reset value: 0x0101

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_PAGE_EN | Reserved | REMOTE_FAULT_EN | EXTENED_NEXT_PAGE | ASYM_PAUSE | PAUSE | 100BASE_T4 | 100BASETX_FULL_DPX | 100BASETX_HALF_DPX | 10BASE-T_FULL_DPX | 10BASE-T_HALF_DPX | SLCT_FLD[4:0] | | | | |
| rw | | rw | rw | rw | rw | r | rw | rw | rw | rw | r | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | NEXT_PAGE_EN | Next Page Enable<br>1: Set to use Next Page<br>0: Not to use Next Page |
| 14 | Reserved | Must be kept at reset value |
| 13 | REMOTE_FAULT_EN | Remote Fault Detection Enable<br>1: Auto Negotiation Fault Detected<br>0: No Remote Fault |
| 12 | EXTENED_NEXT_PAGE | Extended Next Page<br>Not supported in the PHY. Should be wrote 0 all the time |
| 11 | ASYM_PAUSE | Asymmetric Pause Capability<br>Technology Ability A6<br>1: Asymmetric Pause capable<br>0: Asymmetric Pause non-capable |
| 10 | PAUSE | Pause Capability<br>Technology Ability A5<br>1: Pause capable<br>0: Pause non-capable |
| 9 | 100BASE_T4 | 100BASE-T4 Capable<br>Not supported in the PHY. Should be wrote 0 all the time |
| 8 | 100BASETX_FULL_DPX | 100BASE-X Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value |

| | | |
|---|---|---|
| | | programmed in ***PHY control Register (PHY MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8(FORCE_DUPLEX), when FORCE_SPEED is 2'b01 and<br>FORCE_DUPLEX is 1'b1, then this bit will be 1'b1 and vice versa |
| 7 | 100BASETX_HALF_DPX | 100BASE-X Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8 (FORCE_DUPLEX). when FORCE_SPEED is 2'b01 and FORCE_DUPLEX is 1'b0, then this bit will be 1'b1 and vice versa |
| 6 | 10BASE-T_FULL_DPX | 10BASE-T Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8 (FORCE_DUPLEX). When FORCE_SPEED is 2'b00 and FORCE_DUPLEX is 1'b1, then this bit will be 1'b1 and vice versa |
| 5 | 10BASE-T_HALF_DPX | 10BASE-T Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disabled, value on this bit will reflected the value programmed in ***PHY control Register (PHY MII_CTL)*** BIT6, BIT13 (FORCE_SPEED) and BIT8 (FORCE_DUPLEX). when FORCE_SPEED is 2'b00 and FORCE_DUPLEX is 1'b0, then this bit will be 1'b1 and vice versa |
| 4:0 | SLCT_FLD[4:0] | Identifies Type of Message<br>Forced to 5'h01 all the time |

### Auto-Negotiation Link Partner Ability Register (AN_LP_ABILITY)

Device Address: 0x7

Address offset: 0x13

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_PAGE | ACKNOWLEDGE | REMOTE_FAULT | EXTENED_NEXT_PAGE | ASYM_PAUSE | PAUSE | 100BASET4 | 100BASETX_FULL_DPX | 100BASETX_HALF_DPX | 10BASE-T_FULL_DPX | 10BASE-T_HALF_DPX | SELECTOR_FIELD[4:0] | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | NEXT_PAGE | Link Partner Next Page Request<br>1: Link Partner is requesting Next Page function |

| | | |
|---|---|---|
| | | 0: Base Page is requested |
| 14 | ACKNOWLEDGE | Link Partner ACKNOWLEDGE Received<br>1: Link partner acknowledge Received Successfully<br>0: Not Received |
| 13 | REMOTE_FAULT | Link Partner Detects Remote Fault<br>1: Auto Negotiation Fault Detected<br>0: No Remote Fault |
| 12 | EXTENED_NEXT_PAGE | Extended Next Page |
| 11 | ASYM_PAUSE | Link Partner Asymmetric Pause Capable<br>Technology Ability A6<br>1: Asymmetric Pause capable<br>0: Asymmetric Pause non-capable |
| 10 | PAUSE | Link Partner Symmetric Pause Capable<br>Technology Ability A5<br>1: Symmetric Pause capable<br>0: Symmetric Pause non-capable |
| 9 | 100BASET4 | Technology Ability A4<br>Link Partner 100BASE-T4 Capable |
| 8 | 100BASETX_FULL_DPX | Link Partner 100BASE-X Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |
| 7 | 100BASETX_HALF_DPX | Link Partner 100BASE-X Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |
| 6 | 10BASE-T_FULL_DPX | Link Partner 10BASE-T Full Duplex Capable<br>1: Capable of Full Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |
| 5 | 10BASE-T_HALF_DPX | Link Partner 10BASE-T Half Duplex Capable<br>1: Capable of Half Duplex<br>0: Not Capable<br>Note: When Auto-Negotiation is disable, value on this bit will be set to 1’b1 all the time |

| | | |
|---|---|---|
| 4:0 | SELECTOR_FIELD[4:0] | Link Partner Identifies Type of Message Should be 5'h01 |

### Auto-Negotiation XNP Transmit Register (AN_XNP_TRANSMIT)

Device Address: 0x7

Address offset: 0x16

Reset value: 0x2001

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_PAGE | Reserved | MESSAGE_PAGE | ACKNOWLEDGE2 | TOGGLE | MSG_UFMT_CODE_FIELD[10:0] | | | | | | | | | | |
| rw | | rw | rw | r | rw | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | NEXT_PAGE | The Last Next Page<br>Indicated whether this is the last next page<br>1: Additional next page will follow<br>0: This is the last next page |
| 14 | Reserved | Must be kept at reset value. |
| 13 | MESSAGE_PAGE | Message Page or Unformatted Page<br>Indicated this is the message page or unformatted page<br>1: Message Page<br>0: Unformatted Page |
| 12 | ACKNOWLEDGE2 | The Ability to Comply with the Message<br>1: Will comply with the message<br>0: Can not comply with the message |
| 11 | TOGGLE | Toggle<br>The toggle bit will calculated by hardware automatically, SW can ignore |
| 10:0 | MSG_UFMT_CODE_FIELD[10:0] | Message/Unformatted Code Field |

### Auto-Negotiation Link Partner XNP Ability Register (AN_LP_XNP_ABILITY)

Device Address: 0x7

Address offset: 0x19

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NP_RX[15:0] | | | | | | | | | | | | | | | |
| r | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 15 | NP_RX[15:0] | Next Page Received from Link Partner |

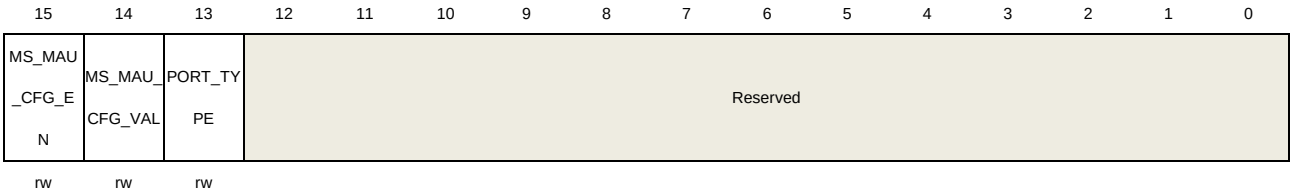
### Master-Slave Control Register (MS_CTL)

Device Address: 0x7

Address offset: 0x20

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | MS_MAU_CFG_EN | Master-Slave Manual Config Enable<br>1: Enable MASTER-SLAVE manual configuration<br>0: Disable MASTER-SLAVE manual configuration |
| 14 | MS_MAU_CFG_VAL | Master-Slave Manual Config Value<br>1: Configure PHY as MASTER<br>0: Configure PHY as SLAVE |
| 13 | PORT_TYPE | Port Type<br>1: Multi-port device<br>0: Single-port device |
| 12:0 | Reserved | Must be kept at reset value. |

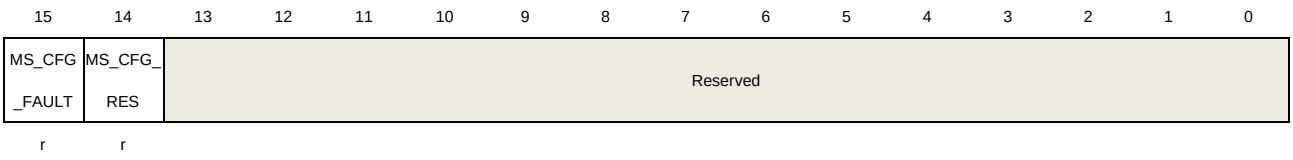
### Master-Slave Resolution Register (MS_STS)

Device Address: 0x7

Address offset: 0x21

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | MS_CFG_FAULT | Master-Slave Config Fault<br>1: MASTER-SLAVE configuration fault detected<br>0: No MASTER-SLAVE configuration fault detected |
| 14 | MS_CFG_RES | Master-Slave Config Resolution |

| | | |
|---|---|---|
| | | 1: Local configuration resolved to MASTER<br>0: Local configuration resolved to SLAVE |
| 13:0 | Reserved | Must be kept at reset value. |

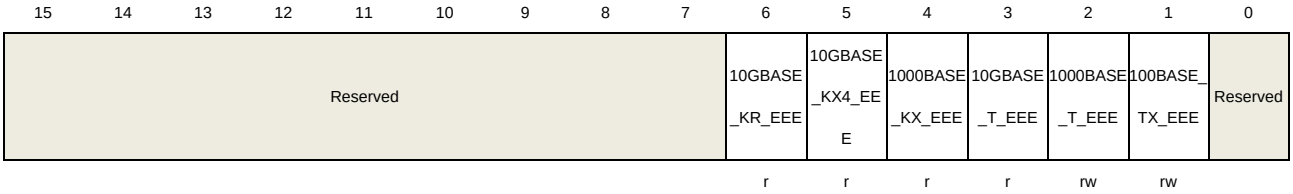
**EEE Advertisement Register (EEE_ADV)**

Device Address: 0x7

Address offset: 0x3C

Reset value: 0x0002

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:7 | Reserved | Must be kept at reset value. |
| 6 | 10GBASE_KR_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 5 | 10GBASE_KX4_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 4 | 1000BASE_KX_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 3 | 10GBASE_T_EEE | 10GBASE-KR EEE<br>Not support, always 0 |
| 2 | 1000BASE_T_EEE | 1000Base-T EEE<br>1: Advertise that the 1000BASE-T has EEE capability<br>0: Do not advertise that the 1000BASE-T has EEE capability |
| 1 | 100BASE_TX_EEE | 100Base-TX EEE<br>1: Advertise that the 100BASE-TX has EEE capability<br>0: Do not advertise that the 100BASE-TX has EEE capability |
| 0 | Reserved | Must be kept at reset value. |

**EEE Link Partner Ability Register (EEE_LP_ABILITY)**

Device Address: 0x7

Address offset: 0x3D

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | 10GBASE_KR_EEE | 10GBASE_KX4_EEE | 1000BASE_KX_EEE | 10GBASE_T_EEE | 1000BASE_T_EEE | 100BASE_TX_EEE | Reserved |
| | | | | | | | | | r | r | r | r | r | r | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:7 | Reserved | Must be kept at reset value. |
| 6 | 10GBASE_KR_EEE | 10GBASE-KR EEE<br>1: Link Partner is advertising EEE capability for 10GBASE-KR<br>0: Link Partner is not advertising EEE capability for 10GBASEKR EEE |
| 5 | 10GBASE_KX4_EEE | 10GBASE-KX4 EEE<br>1: Link Partner is advertising EEE capability for 10GBASEKX4<br>0: Link Partner is not advertising EEE capability for 10GBASEKX4 EEE |
| 4 | 1000BASE_KX_EEE | 1000BASE-KX EEE<br>1: Link Partner is advertising EEE capability for 1000BASE-KX<br>0: Link Partner is not advertising EEE capability for 1000BASE-KX EEE |
| 3 | 10GBASE_T_EEE | 10GBASE-T EEE<br>1: Link Partner is advertising EEE capability for 10GBASE-T<br>0: Link Partner is not advertising EEE capability for 10GBASET EEE |
| 2 | 1000BASE_T_EEE | 1000BASE-T EEE<br>1: Link Partner is advertising EEE capability for 1000BASE-T<br>0: Link Partner is not advertising EEE capability for 1000BASE-T EEE |
| 1 | 100BASE_TX_EEE | 100BASE-TX EEE<br>1: Link Partner is advertising EEE capability for 100BASE-TX<br>0: Link Partner is not advertising EEE capability for 100BASE-TX EEE |
| 0 | Reserved | Must be kept at reset value. |

# 50.9. EtherCAT

## 50.9.1. Overview

The ESC is an EtherCAT® SubDevice Controller (ESC). It takes care of the EtherCAT communication as an interface between the EtherCAT fieldbus and the sub application, The ESC supports a wide range of applications. The EtherCAT controller has 8K bytes of Process Data RAM (PDRAM) and 8 Fieldbus memory management units (FMMUs), each of which performs the task of mapping logical addresses to physical addresses. The EtherCAT SubDevice controller also includes 8 SyncManagers that allow data exchange between the EtherCAT and the native application. The orientation and mode of operation of each SyncManager is configured by the EtherCAT main device. Two working modes are available: buffer mode and mailbox mode. In buffered mode, the µController and EtherCAT main can write devices simultaneously. The buffer in ESC always contains the latest data. If the new data arrives before the old data can be read, the old data will be lost. In mailbox mode, the µController and EtherCAT main access the buffer by shaking hands, ensuring that no data is lost.

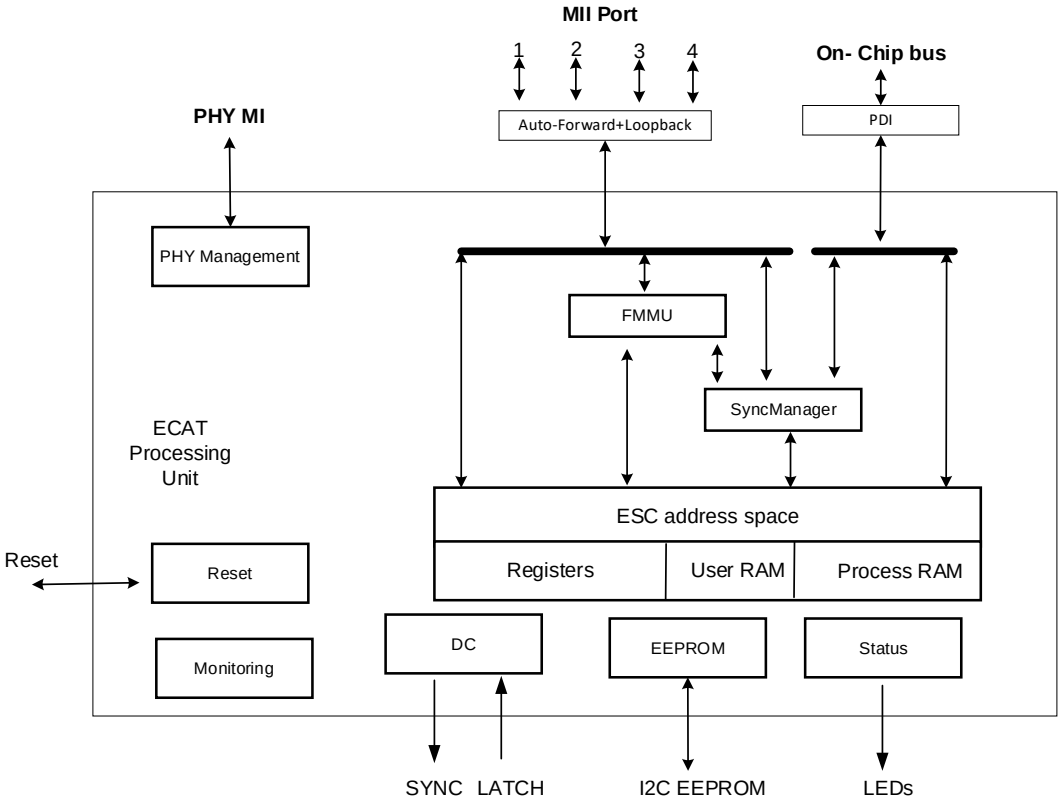
## 50.9.2. Characteristics

- Port support: 2 internal phy port and 1 external MII.
- 8 Fieldbus Memory Management Units (FMMUs).
- 8KB PDRAM
- Distributed clock 64-bit, support allows synchronization with other EtherCAT devices.
- 8 Syncmanager entities.
- DC synchronization deviation is less than 1us.

### Block diagram

The function module of the ESC is shown in ***Figure 50-36. EtherCAT system block diagram***.

**Figure 50-36. EtherCAT system block diagram**




## EtherCAT SubDevice Controller Function Blocks

- EtherCAT Interfaces

The EtherCAT interfaces or ports connect the ESC to other EtherCAT subs and the main. The MAC layer is integral part of the ESC. The physical layer may be Ethernet. For Ethernet ports, internal Ethernet PHYs connect to the MII ports of the ESC. Transmission speed for EtherCAT is fixed to 100 Mbit/s with Full Duplex communication. Link state and communication status are reported to the Monitoring device. ESC uses three ports, port 0/1/2.

- EtherCAT Processing Unit

The EtherCAT Processing Unit (EPU) receives, analyses, and processes the EtherCAT data stream. It is logically located between port 0 and port 3. The main purpose of the EtherCAT Processing unit is to enable and coordinate access to the internal registers and the memory space of the ESC, which can be addressed both from the EtherCAT main and from the local application via the PDI. Data exchange between main and sub application is comparable to a dual-ported memory (process memory), enhanced by special functions e.g. for consistency checking (SyncManager) and data mapping (FMMU). The EtherCAT Processing Units contains the main function blocks of EtherCAT subs besides Auto-Forwarding, Loop-back function, and PDI.

- Auto-Forwarder

The Auto-Forwarder receives the Ethernet frames, performs frame checking and forwards it to the Loop-back function. Time stamps of received frames are generated by the Auto-Forwarder.

- Loop-back function

The Loop-back function forwards Ethernet frames to the next logical port if there is either no link at a port, or if the port is not available, or if the loop is closed for that port. The Loop-back function of port 0 forwards the frames to the EtherCAT Processing Unit. The loop settings can be controlled by the EtherCAT main.

- FMMU

Fieldbus Memory Management Units are used for bitwise mapping of logical addresses to physical addresses of the ESC.

- SyncManager

SyncManagers are responsible for consistent data exchange and mailbox communication between EtherCAT main and subs. The communication direction can be configured for each SyncManager. Read or write transactions may generate events for the EtherCAT main and an attached µController respectively. The SyncManagers are responsible for the main difference between and ESC and a dual-ported memory, because they map addresses to different buffers and block accesses depending on the SyncManager state. This is also a fundamental reason for bandwidth restrictions of the PDI.

- Monitoring

The Monitoring unit contains error counters and watchdogs. The watchdogs are used for observing communication and returning to a safe state in case of an error. Error counters are used for error detection and analysis.

- PHY Management

The PHY Management unit communicates with Ethernet PHYs via the MII management interface. This is either used by the main or by the sub. The MII management interface is used by the ESC itself for optionally restarting auto negotiation after receive errors with the enhanced link detection mechanism, and for the optional MI link detection and configuration feature.

- Distributed Clock

Distributed Clocks (DC) allow for precisely synchronized generation of output signals and input sampling, as well as time stamp generation of events. The synchronization may span the entire EtherCAT network.

- EEPROM

One non-volatile memory is needed for EtherCAT SubDevice Information (ESI) storage, typically an I²C EEPROM.

- Status / LEDs

The Status block provides ESC and application status information. It controls external LEDs like the application RUN LED/ERR LED and port Link/Activity LEDs.

## 50.9.3. Function overview

### Process Data Interface (PDI)

The Process Data Interface (PDI) realizes the connection between sub application and ESC. Several types of PDIs are defined serial and parallel µController interfaces and Digital I/O interfaces. Due to the high dependency between EtherCAT and PDI accesses to memory, registers, and especially SyncManagers, the internal PDI interface can achieve a maximum throughput of approx 12.5 Mbyte/s.

**Table 50-11. PDIs for EtherCAT**

| PDI number (PDI Control register 0x0140) | PDI name |
|---|---|
| 0 | Interface deactivated |
| 128 | On-chip bus |

#### PDI Selection and Configuration

Typically, the PDI selection and configuration is part of the ESC Configuration Area of the SII EEPROM.The ESC has the PDI selected and configured at power-on time. In this case, the ESC Configuration Area should reflect the actual settings, although it is not evaluated by the ESC itself. The PDI is active after reset is released, which enables EEPROM emulation by a µController. Take care of Digital Output signals and DC SyncSignals while the EEPROM is not loaded to achieve proper output behavior.

#### PDI register function acknowledge by write

Some ESC functions are triggered by writing or reading individual byte addresses, SyncManager buffer change or AL event request acknowledge. With an increasing data bus width of the µControllers, this can lead to restrictions or even problems.

Since most µControllers are using byte enable signals for write accesses, there is no restriction for functions which are triggered by writes. But many µControllers are not using the byte enable signals for read accesses, they expect to get a whole data bus width of read data. Reading individual bytes is not possible. This can lead to problems especially by accidentally reading byte addresses which trigger certain ESC functions. Consider a SyncManager buffer area from 0x1000-0x1005. A 32 bit µController application might read the buffer byte-wise. The first access to 0x1000 would open the buffer, and it would also read 0x1001-0x1003. The second access would read 0x1001, and also 0x1000/0x1002-0x1003. The problem occurs when address 0x1004 is to be read, because this would also read 0x1005. The data of 0x1005 is discarded, but the buffer is closed. When the µC reads 0x1005, it will always get 0 – the data seems to be corrupted. A similar issue occurs for DC SyncSignal acknowledging (registers 0x098E and 0x098F). A 32 bit µController would always acknowledge SYNC0 and SYNC1 at the same time, it is not possible to acknowledge them separately.

This problem can be overcome by enabling PDI register function acknowledge by write. In this mode, all functions which are originally triggered by read access are now triggered by corresponding write accesses – which use byte enables and thus can be restricted to certain bytes.

This feature is enabled by IP Core configuration. The current status has to be checked by the µController application in PDI information register 0x014E[0], before using this function.

This feature affects reading of SyncManager buffers and reading of certain registers from PDI side. There is no change to the EtherCAT main side at all. Refer to ***SyncManager*** for SyncManager behavior. The following registers are affected by the PDI register function acknowledge by write feature:

**Table 50-12. Registers affected by PDI register function acknowledge by write**

| Address | name | Trigger function |
|---|---|---|
| any | SyncManager buffer end address | Read SyncManager buffer, then write to buffer end address to acknowledge buffer reading. |
| 0x0120:0x0121 | AL Control | Read 0x0120:0x0121 after AL Control changes, then write to 0x0120 to acknowledge reading. |
| 0x0440 | Watchdog Status Process Data | Read 0x0440, then write to 0x0440 to clear AL event request 0x0220[6] |
| 0x0806+X*16 | SyncManager Activate | Read 0x0806+X*16, then write to 0x0806 (SyncManager 0) only to clear AL event request 0x0220[4] for all SyncManagers |
| 0x098E | SYNC0 Status | Read 0x098E, then write to 0x098E to acknowledge DC Sync0 Status 0x098E[0] |
| 0x098F | SYNC1 Status | Read 0x098E, then write to 0x098E to acknowledge DC Sync1 Status 0x098F[0] |
| 0x09B0:0x09B7 | Latch0 Time Positive Edge | Read 0x09B0:0x09B7, then write to 0x09B0 to clear DC Latch0 Status 0x09AE[0] |
| 0x09B8:0x09BF | Latch0 Time Negative Edge | Read 0x09B8:0x09BF, then write to 0x09B8 to clear DC Latch0 Status 0x09AE[1] |
| 0x09C0:0x09C7 | Latch1 Time Positive Edge | Read 0x09C0:0x09C7, then write to 0x09C0 to clear DC Latch1 Status 0x09AF[0] |
| 0x09C8:0x09CF | Latch1 Time Negative Edge | Read 0x09C8:0x09CF, then write to 0x09C8 to clear DC Latch1 |

| Address | name | Trigger function |
|---|---|---|
| | | Status 0x09AF[1] |

### FMMU

Fieldbus Memory Management Units (FMMU) convert logical addresses into physical addresses by the means of internal address mapping. Thus, FMMUs allow to use logical addressing for data segments that span several sub devices: one datagram addresses data within several arbitrarily distributed EtherCAT. Each FMMU channel maps one continuous logical address space to one continuous physical address space of the sub. The access type supported by an FMMU is configurable to be either read, write, or read/write.

- Restrictions on FMMU Settings

The FMMUs of ESCs are subject to restrictions. The logical address ranges of two FMMUs of the same direction (read or write) in one ESC must be separated by at least 3 logical bytes not configured by any FMMU of the same type, if one of the FMMUs or both use bit-wise mapping (logical start bit $\neq$ 0, logical stop bit $\neq$ 7, or physical start bit $\neq$ 0).

- Additional FMMU Characteristics

  - Each logical address byte can at most be mapped either by one FMMU(read) plus one FMMU(write), or by one FMMU(read/write). If two or more FMMUs (with the same direction – read or write) are configured for the same logical byte, the FMMU with the lower number (lower configuration address space) is used, the other ones are ignored.
  - One or more FMMUs may point to the same physical memory, all of them are used. Collisions cannot occur.
  - It is the same to use one read/write FMMU or two FMMUs – one read, the other one write – for the same logical address.
  - A read/write FMMU cannot be used together with SyncManagers, since independent read and write SyncManagers cannot be configured to use the same (or overlapping) physical address range.
  - Bit-wise reading is supported at any address. Bits which are not mapped to logical addresses are not changed in the EtherCAT datagram. E.g., this allows for mapping bits from several ESCs into the same logical byte.
  - A frame/datagram addressing a logical address space which is not configured in the ESC will not change data in the ESC, and no data from the ESC is placed in the frame/datagram.

### SyncManager

The memory of an ESC can be used for exchanging data between the EtherCAT main and a local application (on a µController attached to the PDI) without any restrictions. Using the memory for communication like this has some drawbacks which are addressed by the SyncManagers inside the ESCs:

- Data consistency is not guaranteed. Semaphores have to be implemented in software

for exchanging data in a coordinated way.

- Data security is not guaranteed. Security mechanisms have to be implemented in software.
- Both EtherCAT main and application have to poll the memory in order to find out when the access of the other side has finished.

SyncManagers enable consistent and secure data exchange between the EtherCAT main and the local application, and they generate interrupts to inform both sides of changes.

SyncManagers are configured by the EtherCAT main. The communication direction is configurable, as well as the communication mode (Buffered Mode and Mailbox Mode). SyncManagers use a buffer located in the memory area for exchanging data. Access to this buffer is controlled by the hardware of the SyncManagers.

A buffer has to be accessed beginning with the start address, otherwise the access is denied. After accessing the start address, the whole buffer can be accessed, even the start address again, either as a whole or in several strokes. A buffer access finishes by accessing the end address, the buffer state changes afterwards and an interrupt or a watchdog trigger pulse is generated (if configured). The end address cannot be accessed twice inside a frame.

Two communication modes are supported by SyncManagers:

- Buffered Mode
  - The buffered mode allows both sides, EtherCAT main and local application, to access the communication buffer at any time. The consumer always gets the latest consistent buffer which was written by the producer, and the producer can always update the content of the buffer. If the buffer is written faster than it is read out, old data will be dropped.
  - The buffered mode is typically used for cyclic process data.
- Mailbox Mode
  - The mailbox mode implements a handshake mechanism for data exchange, so that no data will be lost. Each side, EtherCAT main or local application, will get access to the buffer only after the other side has finished its access. At first, the producer writes to the buffer. Then, the buffer is locked for writing until the consumer has read it out. Afterwards, the producer has write access again, while the buffer is locked for the consumer.
  - The mailbox mode is typically used for application layer protocols.

The SyncManagers accept buffer changes caused by the main only if the FCS of the frame is correct, thus, buffer changes take effect shortly after the end of the frame.

The configuration registers for SyncManagers are located beginning at register address 0x0800.

### Distributed Clocks

The Distributed Clocks (DC) unit of EtherCAT SubDevice controllers supports the following features:

- Clock synchronization between the subs (and the main)
- Generation of synchronous output signals (SyncSignals)
- Precise time stamping of input events (LatchSignals)
- Generation of synchronous interrupts
- Synchronous Digital Output updates
- Synchronous Digital Input sampling

The device supports 64-bit distributed clocks as detailed in the following sub-sections.

The EtherCAT provides two input pins (SYNC and LATCH) which are used for time stamping of external events. Both rising edge and falling edge time stamps are recorded. These pins are shared with the SYNC0 and LATCH0 output pins, respectively, which are used to indicate the occurrence of time events. The functions of the SYNC/ SYNC0 and LATCH /LATCH0 pins are determined by the SYNC0/LATCH0 Configuration and SYNC/LATCH Configuration bits of the Sync/Latch PDI Configuration Register, respectively.

When set for SYNC0/LATCH0 functionality, the output type (Push-Pull vs. Open Drain/Source) and output polarity are determined by the SYNC0 Output Driver/Polarity and LATCH0 Output Driver/Polarity bits of the Sync/Latch PDI Configuration Register.

### EtherCAT State Machine

The EtherCAT State machine (ESM) is responsible for the coordination of main and sub

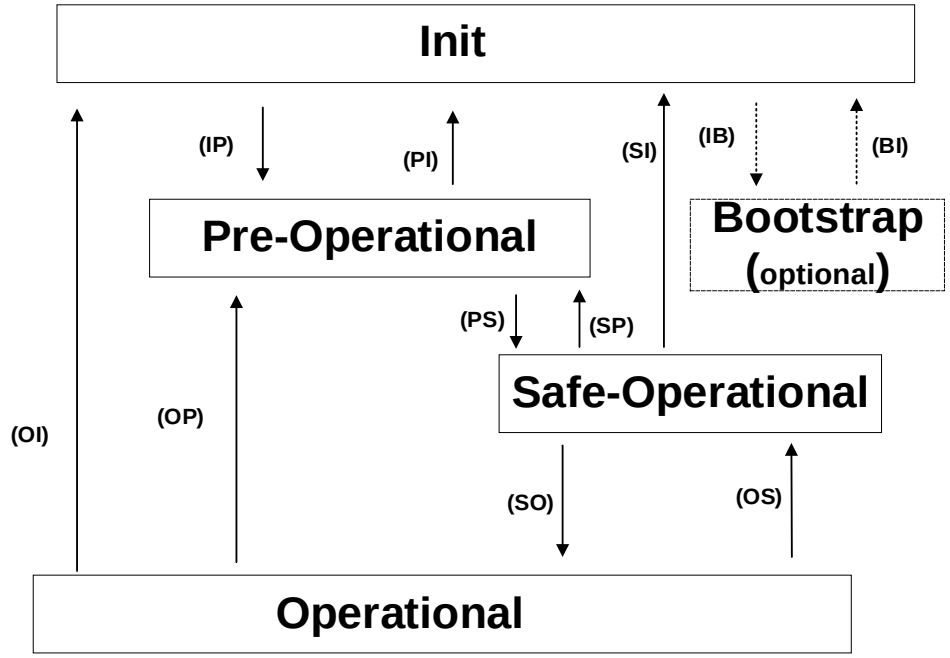
applications at start up and during operation. State changes are typically initiated by requests of the main. They are acknowledged by the local application after the associated operations have been executed. Unsolicited state changes of the local application are also possible.

There are four states an EtherCAT SubDevice shall support, plus one optional state:

- Init (state after Reset)
- Pre-Operational
- Safe-Operational
- Operational
- Bootstrap (optional)

The states and the allowed state changes are shown in ***Figure 50-37. EtherCAT State Machine***:

**Figure 50-37. EtherCAT State Machine**




**NOTE:** Not all state changes are possible, the transition from 'Init' to 'Operational' requires the following sequence: Init -> Pre-Operational -> Save-Operational -> Operational.

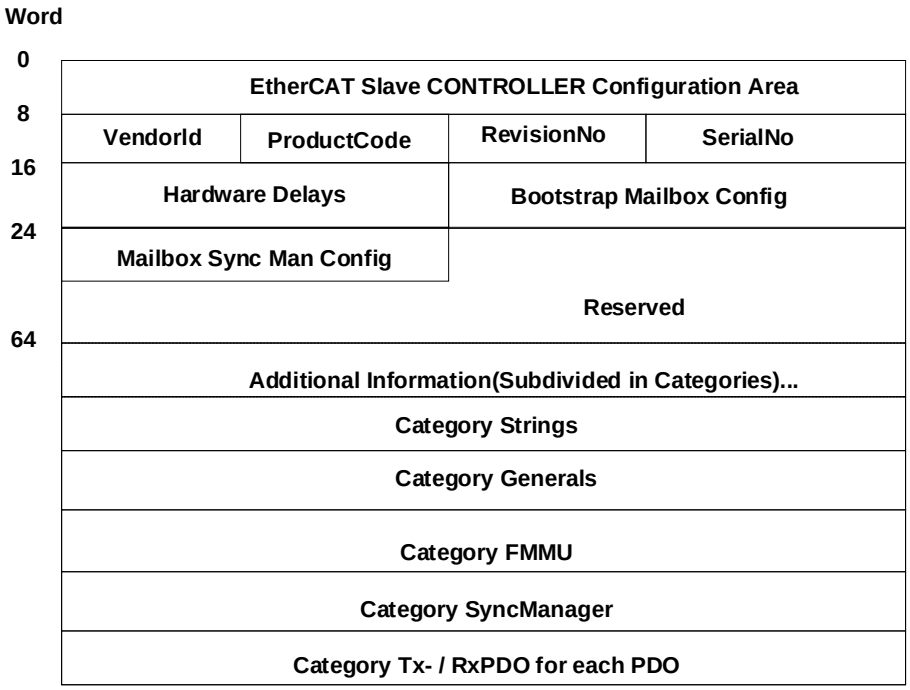
Each state defines required services. Before a state change is confirmed by the sub all services required for the requested state have to be provided or stopped respectively.

### EEPROM

EtherCAT SubDevice controllers use a mandatory RAM (typically a serial EEPROM with I²C interface) to store EtherCAT SubDevice Information (ESI). EEPROM sizes from 1 Kbit up to 4 Mbit are supported, depending on the ESC.

The EEPROM structure is shown in ***Figure 50-38. EEPROM Layout*** , the ESI uses word addressing.

**Figure 50-38. EEPROM Layout**




At least the information stored in the address range from word 0 to 63 (0x00 to 0x3F) is mandatory, as well as the general category (absolute minimum EEPROM size is 2Kbit, complex devices with many categories should be equipped with 32 Kbit EEPROMs or larger). The ESC Configuration area is used by the ESC for configuration. All other parts are used by the main or the local application.

## REST

The EtherCAT module provides two registers, ESC_RESET_ECAT and ESC_RESET_PDI, which can be accessed by the EtherCAT main station and sub station respectively
To trigger a reset request.

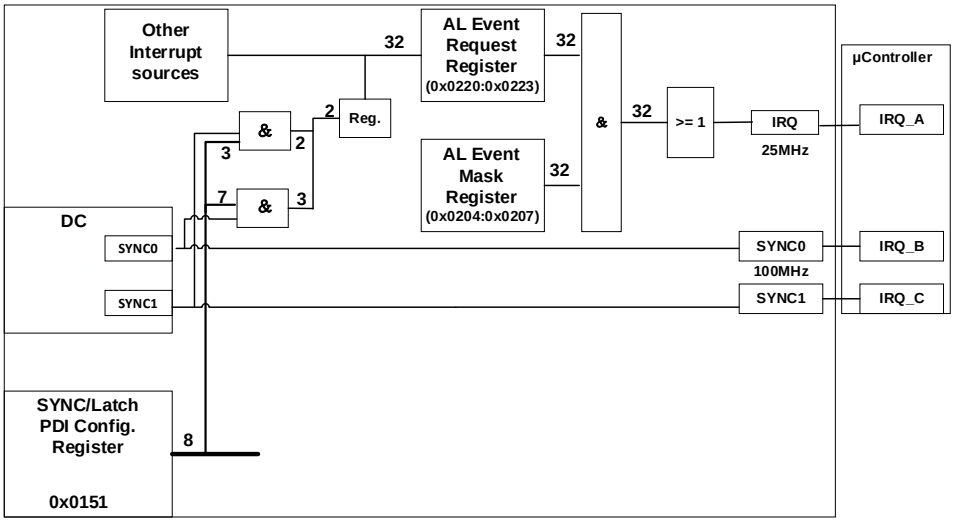
## Interrupts

EtherCAT support two types of interrupts: AL Event Requests targeted at a µController, and EtherCAT event requests targeted at the EtherCAT main. Additionally, the Distributed Clocks SyncSignals can be used as interrupts for a µController as well.

### AL Event Request (PDI Interrupt)

AL Event Requests can be signaled to a µController using the PDI Interrupt Request signal (IRQ/SPI_IRQ, etc.). for IRQ generation, the AL Event Request register (0x0220:0x0223) is combined with the AL Event Mask register (0x0204:0x0207) using a logical AND operation, then all resulting bits are combined (logical OR) into one interrupt signal. The output driver characteristics of the IRQ signal are configurable using the SYNC/LATCH PDI configuration register (0x0151). The AL Event Mask register allows for selecting the interrupts which are relevant for the µController and handled by the application.

**Figure 50-39. PDI Interrupt Masking and interrupt signals**




The DC SyncSignals can be used for interrupt generation in two ways:

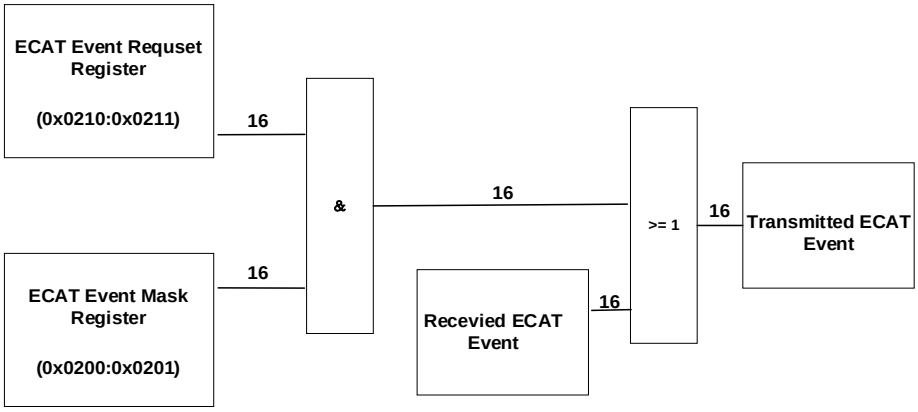
- The DC SYNC signals are mapped into the AL Event Request Register (configured with SYNC/LATCH PDI Configuration register 0x0151.3/7). In this case, all interrupts from the ESC to the µController are combined into one IRQ signal, and the Distributed Clocks LATCH0/1 inputs can still be used. The IRQ signal has a jitter of ~40 ns.
- The DC SyncSignals are directly connected to µController interrupt inputs. The µController can react on DC SyncSignal interrupts faster (without reading AL Request register), but it needs more interrupt inputs. The jitter of the SyncSignals is ~12 ns. The DC Latch functions are only available for one Latch input or not at all (if both DC SYNC outputs are used).

### ECAT Event Request (ECAT Interrupt)

ECAT event requests are used to inform the EtherCAT main of sub events. ECAT events make use of the IRQ field inside EtherCAT datagrams. The ECAT Event Request register (0x0210:0x0211) is combined with the ECAT Event Mask register (0x0200:0x0201) using a logical AND operation. The resulting interrupt bits are combined with the incoming ECAT IRQ field using a logical OR operation, and written into the outgoing ECAT IRQ field. The ECAT Event Mask register allows for selecting the interrupts which are relevant for the EtherCAT main and handled by the main application.

**NOTE:** The main cannot distinguish which sub (or even more than one) was the origin of an interrupt.

**Figure 50-40. EtherCAT Interrupt Masking**




### Clearing Interrupts Accidentally

Event request registers and register actions which clear interrupts are intended to be accessed independently, i.e., with separate EtherCAT frames or separate PDI accesses. Otherwise it may happen that interrupts and/or data are missed.

### LED

EtherCAT SubDevice controllers support LED(RUNLED) regarding link state and AL status. The LED output of an ESC is controlled by the AL status register (0x0130) and supports the following states, which are automatically translated into blink codes.

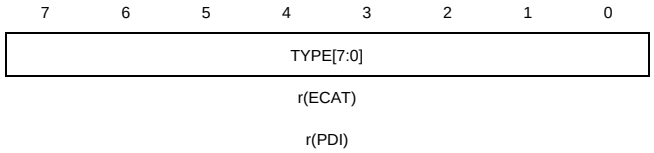
The EtherCAT Core configuration provides for direct control of the RUN LED via the RUN LED Override Register.

## 50.9.4. ESC Register definition

### ESC Type register (ESC_TYPE)

Address Offset: 0x0000
Reset value: 0xBC

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



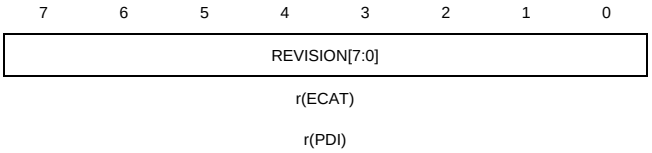

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | TYPE[7:0] | Type of EtherCAT controller<br>0xBC: GDSCN |

### ESC Revision register (ESC_REVISION)

Address Offset: 0x0001

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



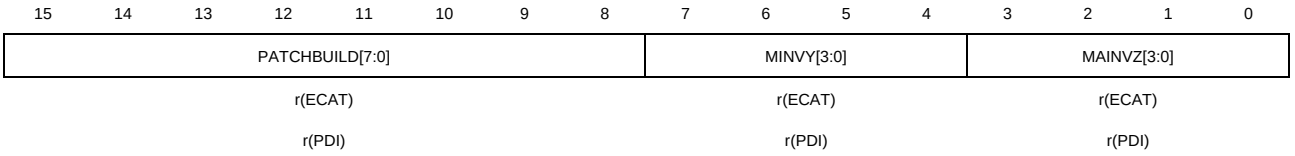

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | REVISION[7:0] | Revision of EtherCAT controller. |

### ESC Build register (ESC_BUILD)

Address Offset: 0x0002

Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



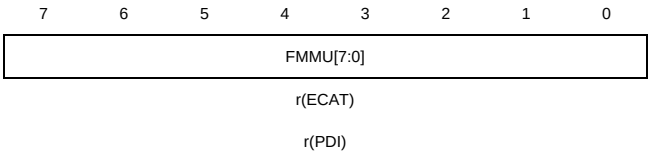

| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | PATCHBUILD[7:0] | patch level / development build:<br>0x00: original release<br>0x01-0x0F: patch level of original release<br>0x10-0xFF: development build |
| 7:4 | MINVY[3:0] | minor version Y |
| 3:0 | MAINVZ[3:0] | maintenance version Z |

### ESC FMMU Numbers register (ESC_FMMUS)

Address Offset: 0x0004

Reset value: 0x08

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



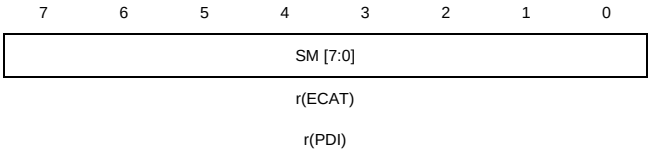
| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | FMMU[7:0] | This field details the number of supported FMMU channels (or entities) of the EtherCAT SubDevice controller. The device provides 8. |

### ESC SyncManagers Numbers register (ESC_SYNCMS)

Address Offset: 0x0005

Reset value: 0x08

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



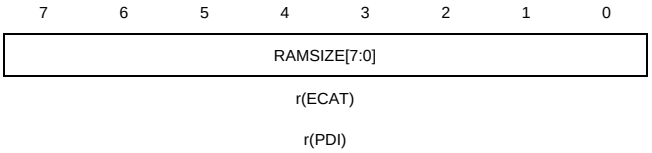

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | SM[7:0] | Number of supported SyncManager entities, The device provides 8. |

### ESC RAM size register (ESC_RAMSIZE)

Address Offset: 0x0006

Reset value: 0x08

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



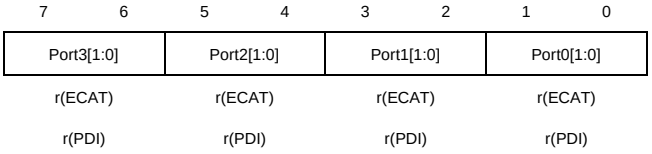

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | RAMSIZE[7:0] | Process Data RAM size supported in 8Kbyte |

### ESC Port Descriptor register (ESC_PORT_DESC)

Address Offset: 0x0007

Reset value: 0x3F

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7:6 | Port3[1:0] | Port 3 Configuration |

| | | |
|---|---|---|
| | | This field details the Port 3 configuration.<br>00: Not implemented<br>01: Not configured<br>10: Reserved<br>11: MII |
| 5:4 | Port2[1:0] | Port 2 Configuration<br>This field details the Port 2configuration.<br>00: Not implemented<br>01: Not configured<br>10: Reserved<br>11: MII |
| 3:2 | Port1[1:0] | Port 1 Configuration<br>This field details the Port 1 configuration.<br>00: Not implemented<br>01: Not configured<br>10: Reserved<br>11: MII |
| 1:0 | Port0[1:0] | Port 0 Configuration<br>This field details the Port 0 configuration.<br>00: Not implemented<br>01: Not configured<br>10: Reserved<br>11: MII |

### ESC Features supporter register (ESC_FEATURE)

Address Offset: 0x0008
Reset value: 0x01CC

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | FS11 | FS10 | FS9 | FS8 | FS7 | FS6 | Reserved | | FS3 | FS2 | FS1 | FS0 |
| | | | | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | | | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) |
| | | | | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | | | r(PDI) | r(PDI) | r(PDI) | r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:12 | Reserved | Must be kept at reset value. |
| 11 | FS11 | Fixed FMMU/SyncManager configuration:<br>0: Variable configuration<br>1: Fixed configuration |
| 10 | FS10 | EtherCAT read/write command support (BRW, APRW, FPRW):<br>0: Supported |

| | | |
|---|---|---|
| | | 1: Not supported |
| 9 | FS9 | EtherCAT LRW command support:<br>0: Supported<br>1: Not supported |
| 8 | FS8 | Enhanced DC SYNC Activation:<br>0: Not available<br>1: Available NOTE: This feature refers to registers ***ESC Register Activation register (ESC_REG_ACT)*** and ***ESC Activation Status register (ESC_ACT_STAT)***. |
| 7 | FS7 | Separate Handling of FCS Errors:<br>0: Not supported<br>1: Supported, frames with wrong FCS and additional nibble will be counted separately in Forwarded RX Error Counter |
| 6 | FS6 | Enhanced Link Detection MII:<br>0: Not available<br>1: Available |
| 5:4 | Reserved | Must be kept at reset value. |
| 3 | FS3 | Distributed Clocks (width):<br>0: 32 bit<br>1: 64 bit |
| 2 | FS2 | Distributed Clocks:<br>0: Not available<br>1: Available |
| 1 | FS1 | Unused register access:<br>0: allowed<br>1: not supported |
| 0 | FS0 | FMMU Operation<br>0: Bit oriented<br>1: Byte oriented |

### ESC Configured station address register (ESC_STATION_ADDR)

Address Offset: 0x0010

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

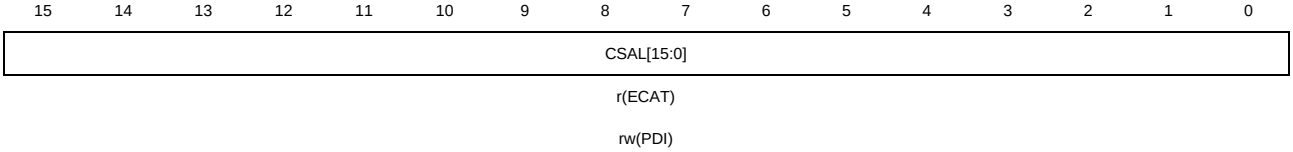
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSAD[15:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CSAD[15:0] | Address used for node addressing (FPRD/FPWR/FPRW/FRMW commands). |

### ESC Configured station Alias register (ESC_STATION_ALS)

Address Offset: 0x0012
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

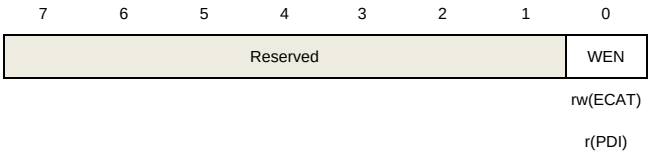



| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | CSAL[15:0] | Alias Address used for node addressing (FPRD/FPWR/FPRW/FRMW commands). The use of this alias is activated by Register ***ESC DL Control register (ESC_DL_CTL)*** Bit [24].<br><br>NOTE: EEPROM value is only transferred into this register at first EEPROM load after power-on or reset. |

### Write Enable register (ESC_REG_WEN)

Address Offset: 0x0020
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

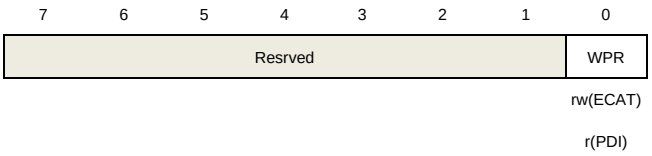



| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | WEN | If register write protection is enabled, this register has to be written in the same Ethernet frame (value does not matter) before other writes to this station are allowed. This bit is self-clearing at the beginning of the next frame (SOF), or if Register Write Protection is disabled. |

### ESC Write Protection register (ESC_REG_WP)

Address Offset: 0x0021
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



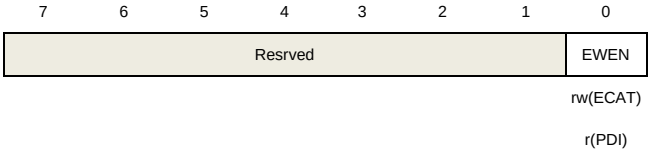

| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | WPR | Register write protection:<br>0: Protection disabled<br>1: Protection enabled<br><br>Registers 0x0000:0x0F7F are write-protected, except for 0x0020 and 0x0030. |

### ESC Write Enable register (ESC_WEN)

Address Offset: 0x0030

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



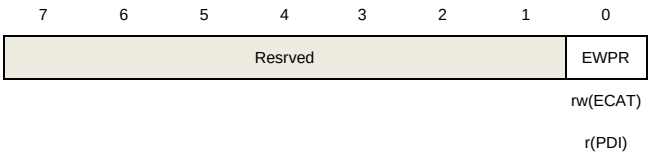

| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | EWEN | If ESC write protection is enabled, this register has to be written in the same Ethernet frame (value does not matter) before other writes to this station are allowed. This bit is self-clearing at the beginning of the next frame (SOF), or if ESC Write Protection is disabled. |

### ESC Write Protection register (ESC_WP)

Address Offset: 0x0031

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |

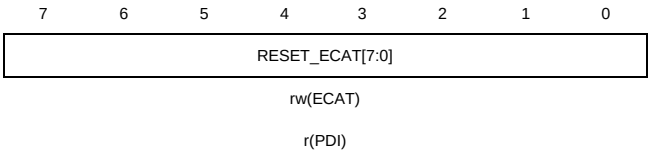
| | | |
|---|---|---|
| 0 | EWPR | Write protect:<br>0: Protection disabled<br>1: Protection enabled<br>All areas are write-protected, except for 0x0030. |

### ESC Reset register (ESC_RST_ECAT)

Address Offset: 0x0040
Reset value: 0x0

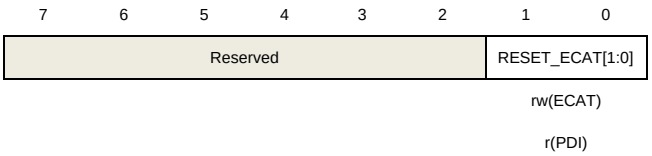
This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

**Write:**




| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | RESET_ECAT[7:0] | A reset is asserted after writing the reset sequence 0x52 (‘R’), 0x45 (‘E’) and 0x53 (‘S’) in this register with 3 consecutive frames. Any other frame which does not continue the sequence by writing the next expected value will cancel the reset procedure. |

**Read:**



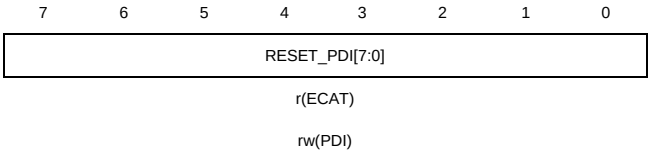

| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1:0 | RESET_ECAT[1:0] | Progress of the reset procedure:<br>00: initial/reset state<br>01: after writing 0x52 (‘R’), when previous state was 00<br>10: after writing 0x45 (‘E’), when previous state was 01<br>11: after writing 0x53 (‘S’), when previous state was 10.<br>This value must not be observed because the ESC enters reset when this state is reached, resulting in state 00. |

### ESC Reset PDI register (ESC_RST_PDI)

Address Offset: 0x0041
Reset value: 0x0

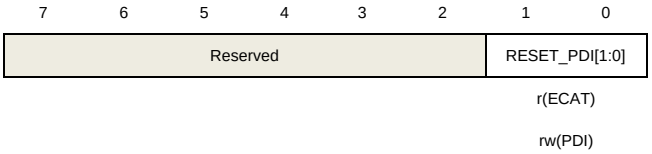
This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

**Write:**




| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | RESET_PDI[7:0] | A reset is asserted after writing the reset sequence 0x52 (‘R’), 0x45 (‘E’) and 0x53 (‘S’) in this register with 3 consecutive commands. Any other command which does not continue the sequence by writing the next expected value will cancel the reset procedure. |

**Read:**



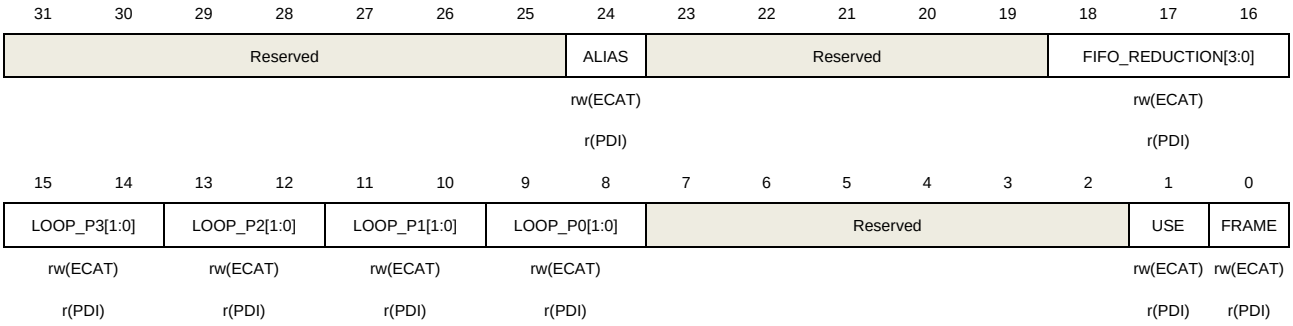

| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1:0 | RESET_PDI[1:0] | Progress of the reset procedure:<br>00: initial/reset state<br>01: after writing 0x52 (‘R’), when previous state was 00<br>10: after writing 0x45 (‘E’), when previous state was 01<br>11: after writing 0x53 (‘S’), when previous state was 10.<br>This value must not be observed because the ESC enters reset when this state is reached, resulting in state 00. |

### ESC DL Control register (ESC_DL_CTL)

Address Offset: 0x0100

Reset value: 0x7C001

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|

| | | |
|---|---|---|
| 31:25 | Reserved | Must be kept at reset value. |
| 24 | ALIAS | Station alias:<br>0: Ignore Station Alias<br>1: Alias can be used for all configured address command types (FPRD, FPWR, …) |
| 23:19 | Reserved | Must be kept at reset value. |
| 18:16 | FIFO_REDUCTION[2:0] | RX FIFO Size (ESC delays start of forwarding until FIFO is at least half full). RX FIFO Size/RX delay reduction :<br>Value: MII:<br>0: -40 ns (-80 ns)<br>1: -40 ns (-80 ns)<br>2: -40 ns<br>3: -40 ns<br>4: no change<br>5: no change<br>6: no change<br>7: default<br>NOTE: EEPROM value is only taken over at first EEPROM load after power-on or reset |
| 15:14 | LOOP_P3[1:0] | Loop Port 3:<br>00: Auto<br>01: Auto Close<br>10: Open<br>11: Closed |
| 13:12 | LOOP_P2[1:0] | Loop Port 2:<br>00: Auto<br>01: Auto Close<br>10: Open<br>11: Closed |
| 11:10 | LOOP_P1[1:0] | Loop Port 1:<br>00: Auto<br>01: Auto Close<br>10: Open<br>11: Closed |
| 9:8 | LOOP_P0[1:0] | Loop Port 0:<br>00: Auto<br>01: Auto Close<br>10: Open<br>11: Closed<br>NOTE: Loop open means sending/receiving over this port is enabled, loop |

| | | |
|---|---|---|
| | | closed means sending/receiving is disabled and frames are forwarded to the next open port internally.<br>Auto: loop closed at link down, opened at link up<br>Auto Close: loop closed at link down, opened with writing 01 again after link up (or receiving a valid Ethernet frame at the closed port)<br>Open: loop open regardless of link state<br>Closed: loop closed regardless of link state |
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | USE | Temporary use of settings in 0x0100:0x0103[8:15]:<br>0: permanent use<br>1: use for about 1 second, then revert to previous settings |
| 0 | FRAME | Forwarding rule:<br>0: Forward non-EtherCAT frames: EtherCAT frames are processed, non-EtherCAT frames are forwarded without processing or modification. The source MAC address is not changed for any frame.<br>1: Destroy non-EtherCAT frames: EtherCAT frames are processed, non-EtherCAT frames are destroyed. The source MAC address is changed by the Processing Unit for every frame (SOURCE_MAC[1] is set to 1 – locally administered address). |

### ESC Physical read/write offset register (ESC_PHYSICAL_RW_OFFSET)

Address Offset: 0x0108
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RW_OFFSET[15:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | RW_OFFSET[15:0] | This register is used for ReadWrite commands in Device Addressing mode (FPRW, APRW, BRW). The internal read address is directly taken from the offset address field of the EtherCAT datagram header, while the internal write address is calculated by adding the Physical Read/Write Offset value to the offset address field. Internal read address = ADR, internal write address = ADR + R/W-Offset |

### ESC DL Status register (ESC_DL_STAT)

Address Offset: 0x0110
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DL_STAT US15 | DL_STAT US14 | DL_STAT US13 | DL_STAT US12 | DL_STAT US1` | DL_STAT US10 | DL_STAT US9 | DL_STAT US8 | DL_STAT US7 | DL_STAT US6 | DL_STAT US5 | DL_STAT US4 | Reserved | DL_STAT US2 | DL_STAT US1 | DL_STAT US0 |
| r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | DL_STATUS15 | Communication on Port 3:<br>0: No stable communication<br>1: Communication established |
| 14 | DL_STATUS14 | Loop Port 3:<br>0: Open<br>1: Closed |
| 13 | DL_STATUS13 | Communication on Port 2:<br>0: No stable communication<br>1: Communication established |
| 12 | DL_STATUS12 | Loop Port 2:<br>0: Open<br>1: Closed |
| 11 | DL_STATUS11 | Communication on Port 1:<br>0: No stable communication<br>1: Communication established |
| 10 | DL_STATUS10 | Loop Port 1:<br>0: Open<br>1: Closed |
| 9 | DL_STATUS9 | Communication on Port 0:<br>0: No stable communication<br>1: Communication established |
| 8 | DL_STATUS8 | Loop Port 0:<br>0: Open<br>1: Closed |
| 7 | DL_STATUS7 | Physical link on Port 3<br>0: No link<br>1: Link detected |
| 6 | DL_STATUS6 | Physical link on Port 2<br>0: No link<br>1: Link detected |
| 5 | DL_STATUS5 | Physical link on Port 1<br>0: No link |

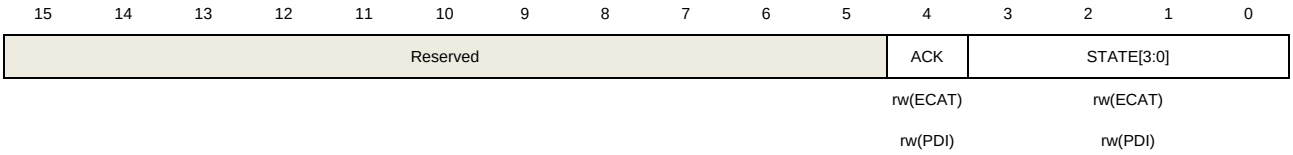
| | | |
|---|---|---|
| | | 1: Link detected |
| 4 | DL_STATUS4 | Physical link on Port 0:<br>0: No link<br>1: Link detected |
| 3 | Reserved | Must be kept at reset value. |
| 2 | DL_STATUS2 | Enhanced Link detection:<br>0: Deactivated for all ports<br>1: Activated for at least one port NOTE: EEPROM value is only transferred into this register at first EEPROM load after power-on or reset |
| 1 | DL_STATUS1 | PDI Watchdog Status:<br>0: Watchdog expired<br>1: Watchdog reloaded |
| 0 | DL_STATUS0 | PDI operational/EEPROM loaded correctly:<br>0: EEPROM not loaded, PDI not operational (no access to Process Data RAM)<br>1: EEPROM loaded correctly, PDI operational (access to Process Data RAM) |

### ESC AL Control register (ESC_AL_CTL)

Address Offset: 0x0120

Reset value: 0x1

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:5 | Reserved | Must be kept at reset value. |
| 4 | ACK | Error Ind Ack:<br>0: No Ack of Error Ind in AL status register<br>1: Ack of Error Ind in AL status register |
| 3:0 | STATE[3:0] | Initiate State Transition of the Device State Machine:<br>1: Request Init State<br>3: Request Bootstrap State<br>2: Request Pre-Operational State<br>4: Request Safe-Operational State<br>8: Request Operational State<br>Note: PDI register function acknowledge by Write command is disabled: Reading AL Control from PDI clears AL Event Request 0x0220[0]. Writing to this register from PDI is not possible. PDI register function acknowledge by Write command is enabled: Writing AL Control from PDI clears AL Event Request 0x0220[0]. Writing |

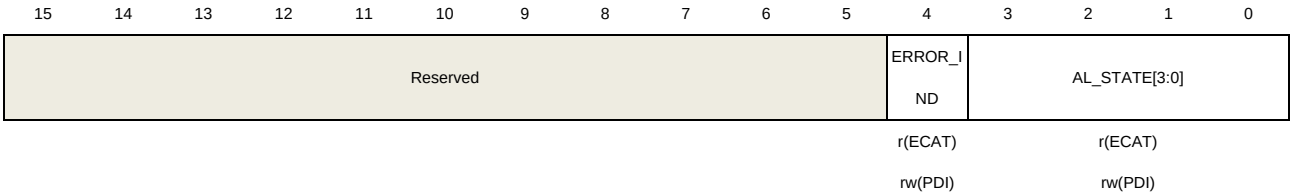
to this register from PDI is possible; write value is ignored (write 0).

**NOTE:** AL Control register behaves like a mailbox if Device Emulation is off (0x0141[0]=0): The PDI has to read/write* the AL Control register after ECAT has written it. Otherwise ECAT cannot write again to the AL Control register. After Reset, AL Control register can be written by ECAT. (Regarding mailbox functionality, both low and high byte of the AL Control register trigger read/write functions, e.g., reading 0x0121 is sufficient to make this register writable again) If Device Emulation is on, the AL Control register can always be written, its content is copied to the AL Status register.

### ESC AL Status register (ESC_AL_STAT)

Address Offset: 0x0130
Reset value: 0x1

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



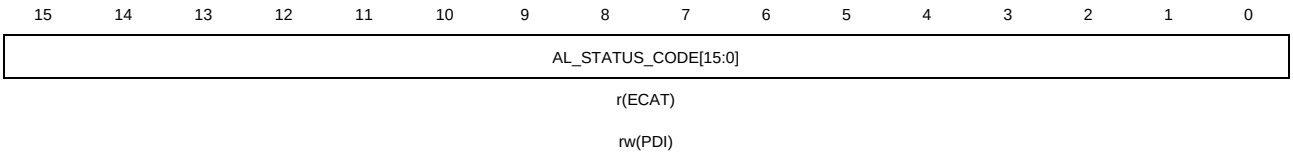

| Bits | Fields | Descriptions |
|---|---|---|
| 15:5 | Reserved | Must be kept at reset value. |
| 4 | ERROR_IND | Error Ind:<br>0: Device is in State as requested or Flag cleared by command<br>1: Device has not entered requested State or changed State as result of a local action |
| 3:0 | AL_STATE[3:0] | Actual State of the Device State Machine:<br>1: Init State<br>3: Bootstrap State<br>2: Pre-Operational State<br>4: Safe-Operational State<br>8: Operational State |

**NOTE:** AL Status register is only writable from PDI if Device Emulation is off (0x0141[0]=0), otherwise AL Status register will reflect AL Control register values. Avoid reading AL Status register from PDI.

### ESC AL Status Code register (ESC_AL_STAT_CODE)

Address Offset: 0x0134
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



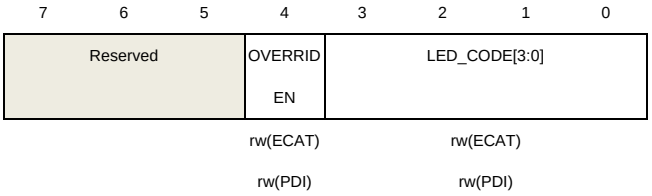

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | AL_STATUS_CODE[15:0] | AL Status Code |

### ESC RUN LED Override register (ESC_RUN_LED_OVRD)

Address Offset: 0x0138

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions | |
|---|---|---|---|
| 7:5 | Reserved | Must be kept at reset value. | |
| 4 | OVERRIDEN | Enable Override:<br>0: Override disabled<br>1: Override enabled | |
| 3:0 | LED_CODE[3:0] | LED code: | AL Status: |
| | | 0x0: Off | Init 1 |
| | | 0x1-0xC: Flash1x – 12x | SafeOp 4 |
| | | 0xD: Blinking | PreOp 2 |
| | | 0xE: Flickering | Bootstrap 3 |
| | | 0xF: On | Operational 8 |

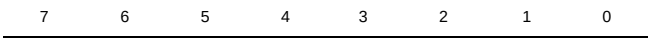
**NOTE:** Changes to AL Status register (0x0130) with valid values will disable RUN LED Override (0x0138[4]=0). The value read in this register always reflects current LED output.

### ESC ERR LED Override register (ESC_ERR_LED_OVRD)

Address Offset: 0x0139

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| Reserved | OVERRIDEN | LED_CODE[3:0] |
|---|---|---|
| | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:5 | Reserved | Must be kept at reset value. |
| 4 | OVERRIDEN | Enable Override:<br>0: Override disabled<br>1: Override enabled |
| 3:0 | LED_CODE[3:0] | LED code:<br>0x0: Off<br>0x1-0xC: Flash 1x – 12x<br>0xD: Blinking<br>0xE: Flickering<br>0xF: On |

**NOTE:** New error conditions will disable ERR LED Override (0x0139[4]=0). The value read in this register always reflects current LED output.

### ESC PDI Control register (ESC_PDI_CTL)

Address Offset: 0x0140
Reset value: 0x80

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PDI_CTL[7:0] | | | | | | | |
| r(ECAT) r(PDI) | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | PDI_CTL[7:0] | Process data interface:<br>0x80: PDI Select SPI<br>Others: Reserved |

### ESC Configuration register (ESC_CFG)

Address Offset: 0x0141
Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EH_LINK_P3 | EH_LINK_P2 | EH_LINK_P1 | EH_LINK_P0 | CLK_LATCH_EN | CLK_SYNC_EN | EH_LINK_ALL | CTL_AL_STATUS |

| r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |
|---|---|---|---|---|---|---|---|

| Bits | Fields | Descriptions |
|---|---|---|
| 7 | EH_LINK_P3 | Enhanced Link port 3(default 1, later EEPROM word 0):<br>0: disabled (if bit 1=0)<br>1: enabled |
| 6 | EH_LINK_P2 | Enhanced Link port 2(default 1, later EEPROM word 0):<br>0: disabled (if bit 1=0)<br>1: enabled |
| 5 | EH_LINK_P1 | Enhanced Link port 1(default 1, later EEPROM word 0):<br>0: disabled (if bit 1=0)<br>1: enabled |
| 4 | EH_LINK_P0 | Enhanced Link port 0(default 1, later EEPROM word 0):<br>0: disabled (if bit 1=0)<br>1: enabled |
| 3 | CLK_LATCH_EN | Distributed Clocks Latch In Unit:<br>0: disabled (power saving)<br>1: enabled |
| 2 | CLK_SYNC_EN | Distributed Clocks SYNC Out Unit:<br>0: disabled (power saving)<br>1: enabled |
| 1 | EH_LINK_ALL | Enhanced Link detection all ports(default 1, later EEPROM word 0):<br>0: disabled (if bits [7:4]=0)<br>1: enabled at all ports (overrides bits [7:4]) |
| 0 | CTL_AL_STATUS | Device emulation (control of AL status) (default 0)<br>0: AL status register has to be set by PDI<br>1: AL status register will be set to value written to AL control register<br>NOTE: Reset value is 1 with Digital I/O PDI，On-chip bus Others: 0。 |

**NOTE:** EEPROM values of bits 1, 4, 5, 6, and 7 are only transferred into this register at first EEPROM load after power-on or reset.

### ESC PDI Information register (ESC_PDI_INFO)

Address Offset: 0x014E

Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | | | | | CONFIGURE_EN | ACTIVE_EN | LOAD_EN | PDI_WREN |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:4 | Reserved | Must be kept at reset value. |
| 3 | CONFIGURE_EN | PDI configuration invalid(default 0):<br>0: PDI configuration ok<br>1: PDI configuration invalid |
| 2 | ACTIVE_EN | PDI active(default 0):<br>0: PDI not active<br>1: PDI active |
| 1 | LOAD_EN | ESC configuration area loaded from EEPROM(default 0):<br>0: not loaded<br>1: loaded |
| 0 | PDI_WREN | PDI function acknowledge by write:<br>0: Disabled<br>1: Enabled<br>Note: Reset value depends on configuration |

### ESC PDI configuration register (ESC_PDI_CFG)

Address Offset: 0x0150

Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 7 6 | 5 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| OUT_DATA[1:0] | IN_DATA[1:0] | WATCHDOG | MODE | OUTVALID_MOD | POLARITY |
| r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:6 | OUT_DATA[1:0] | Output DATA is updated at<br>00: End of Frame<br>01: reserved<br>10: DC SYNC0 event<br>11: DC SYNC1 event<br>If 0x0150[1]=1, output DATA is updated at Process Data Watchdog trigger event (0x0150[7:6] are ignored) |
| 5:4 | IN_DATA[1:0] | Input DATA is sampled at<br>00: Start of Frame2<br>01: Rising edge of LATCH_IN |

| | | |
|---|---|---|
| | | 10: DC SYNC0 event<br>11: DC SYNC1 event |
| 3 | WATCHDOG | Watchdog behavior(default 0):<br>0: Outputs are reset immediately after watchdog expires<br>1: Outputs are reset with next output event that follows watchdog expiration |
| 2 | MODE | Unidirectional/Bidirectional mode(default 0):<br>0: Unidirectional mode: input/output direction of pins configured individually<br>1: Bidirectional mode: all I/O pins are bidirectional, direction configuration is ignored |
| 1 | OUTVALID_MOD | OUTVALID mode(default 0):<br>0: Output event signaling<br>1: Process Data Watchdog trigger (WD_TRIG) signaling on OUTVALID pin (see SyncManager). Output data is updated if watchdog is triggered. Overrides 0x0150[7:6] |
| 0 | POLARITY | OUTVALID polarity(default 0):<br>0: Active high<br>1: Active low |

**NOTE:** all the bit can be configured via EEPROM.

### ESC Sync/Latch configuration register (ESC_SL_CFG)

Address Offset: 0x0151
Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SYNC1_MAP | SL1_CONFIG | SYNC1_POLARITY[0:1] | | SYNC0_MAP | SL0_CONFIG | SYNC1_POLARITY[1:0] | |
| r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | |

| Bits | Fields | Descriptions |
|---|---|---|
| 7 | SYNC1_MAP | SYNC1 mapped to AL Event Request Register 0x0220[3]:<br>0: Disabled<br>1: Enabled |
| 6 | SL1_CONFIG | SYNC1/LATCH1 Configuration(default 1):<br>0: LATCH1 Input<br>1: SYNC1 Output<br>NOTE: The ESC has concurrent SYNC[1:0] outputs and LATCH[1:0] inputs, independent of this configuration |
| 5:4 | SYNC1_POLARITY[0:1] | SYNC1 Output Driver/Polarity(default 10):<br>00: Push-Pull (Active Low) |

| | | |
|---|---|---|
| | | 01: Open Drain (Active Low)<br>10: Push-Pull (Active High)<br>11: Open Source (Active High) |
| 3 | SYNC0_MAP | SYNC0 mapped to AL Event Request Register 0x0220[2]:<br>0: Disabled<br>1: Enabled |
| 2 | SL0_CONFIG | SYNC0/LATCH0 Configuration(default 1):<br>0: LATCH0 Input<br>1: SYNC0 Output<br>NOTE: The ESC has concurrent SYNC[1:0] outputs and LATCH[1:0] inputs, independent of this configuration |
| 1:0 | SYNC1_POLARITY[1:0] | SYNC0 Output Driver/Polarity(default 10):<br>00: Push-Pull (Active Low)<br>01: Open Drain (Active Low)<br>10: Push-Pull (Active High)<br>11: Open Source (Active High) |

### ESC PDI extended configuration register (ESC_PDI_EXT_CFG)

Address Offset: 0x0152

Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DIR15 | DIR15 | DIR15 | DIR15 | DIR15 | DIR15 | DIR15 | DIR8 | DIR7 | DIR6 | DIR5 | DIR4 | DIR3 | DIR2 | DIR1 | DIR0 |
| r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) | r(ECAT) |
| r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) | r(PDI) |

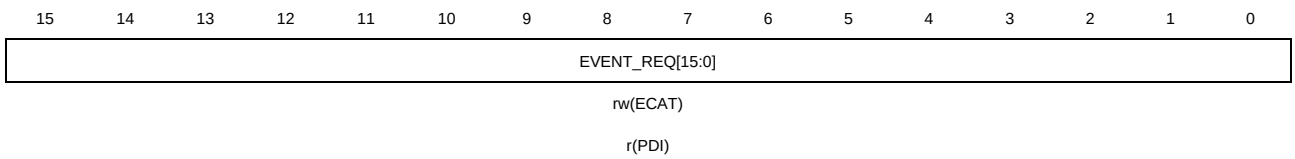
| Bits | Fields | Descriptions |
|---|---|---|
| 15 | DIR15 | Direction of I/O[31:30] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 14 | DIR14 | Direction of I/O[29:28] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 13 | DIR13 | Direction of I/O[27:26] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 12 | DIR12 | Direction of I/O[25:24] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 11 | DIR11 | Direction of I/O[23:22] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 10 | DIR10 | Direction of I/O[21:20] configured in pairs as inputs or outputs: |

| | | |
|---|---|---|
| | | Refer to DIR0 description |
| 9 | DIR9 | Direction of I/O[19:18] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 8 | DIR8 | Direction of I/O[17:16] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 7 | DIR7 | Direction of I/O[15:14] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 6 | DIR6 | Direction of I/O[13:12] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 5 | DIR5 | Direction of I/O[11:10] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 4 | DIR4 | Direction of I/O[9:8] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 3 | DIR3 | Direction of I/O[7:6] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 2 | DIR2 | Direction of I/O[5:4] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 1 | DIR1 | Direction of I/O[3:2] configured in pairs as inputs or outputs:<br>Refer to DIR0 description |
| 0 | DIR0 | Direction of I/O[1:0] configured in pairs as inputs or outputs:<br>0: Input<br>1: Output<br>NOTE: Reserved in bidirectional mode, set to 0. Configuration bits for unavailable I/Os are reserved, set EEPROM value to 0. |

### ESC Event Mask register (ESC_EVT_MSK)

Address Offset: 0x0200
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | EVENT_REQ[15:0] | ECAT Event masking of the ECAT Event Request Events for mapping into ECAT event field of EtherCAT frames:<br>0: Corresponding ECAT Event Request register bit is not mapped |

1: Corresponding ECAT Event Request register bit is mapped

### ESC PDI AL Event register (ESC_PDI_AL_EVT)

Address Offset: 0x0204

Reset value: 0x00FFFF0F

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AL_EVENT_REQ[31:16] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| AL_EVENT_REQ[15:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | AL_EVENT_REQ[31:0] | AL Event masking of the AL Event Request register Events for mapping to PDI IRQ signal:<br>0: Corresponding AL Event Request register bit is not mapped<br>1: Corresponding AL Event Request register bit is mapped |

### ESC Event Request register (ESC_EVT_REQ)

Address Offset: 0x0210

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | CHANNEL7 | CHANNEL6 | CHANNEL5 | CHANNEL4 | CHANNEL3 | CHANNEL2 | CHANNEL1 | CHANNEL0 | AL_STATUS | DL_STATUS | Reserved | DC_LATCH |
| | | | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:12 | Reserved | Must be kept at reset value. |
| 11 | CHANNEL7 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 7 event<br>1: Sync Channel 7 event pending |
| 10 | CHANNEL6 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 6 event<br>1: Sync Channel 6 event pending |
| 9 | CHANNEL5 | Mirrors values of each SyncManager Status: |

| | | |
|---|---|---|
| | | 0: No Sync Channel 5 event<br>1: Sync Channel 5 event pending |
| 8 | CHANNEL4 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 4 event<br>1: Sync Channel 4 event pending |
| 7 | CHANNEL3 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 3 event<br>1: Sync Channel 3 event pending |
| 6 | CHANNEL2 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 2 event<br>1: Sync Channel 2 event pending |
| 5 | CHANNEL1 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 1 event<br>1: Sync Channel 1 event pending |
| 4 | CHANNEL0 | Mirrors values of each SyncManager Status:<br>0: No Sync Channel 0 event<br>1: Sync Channel 0 event pending |
| 3 | AL_STATUS | AL Status event:<br>0: No change in AL Status<br>1: AL Status change<br>(Bit is cleared by reading out AL Status 0x0130:0x0131 from ECAT) |
| 2 | DL_STATUS | DL Status event:<br>0: No change in DL Status<br>1: DL Status change<br>(Bit is cleared by reading out DL Status 0x0110:0x0111 from ECAT) |
| 1 | Reserved | Must be kept at reset value. |
| 0 | DC_LATCH | DC Latch event:<br>0: No change on DC Latch Inputs<br>1: At least one change on DC Latch Inputs<br>(Bit is cleared by reading DC Latch event times from ECAT for ECAT-controlled Latch Units, so that Latch 0/1 Status 0x09AE:0x09AF indicates no event) |

### ESC AL Event Request register (ESC_AL_EVT_REQ)

Address Offset: 0x0220

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | SYNC15_INT | SYNC14_INT | SYNC13_INT | SYNC12_INT | SYNC11_INT | SYNC10_INT | SYNC9_INT | SYNC8_INT |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SYNC7_INT | SYNC6_INT | SYNC5_INT | SYNC4_INT | SYNC3_INT | SYNC2_INT | SYNC1_INT | SYNC0_INT | Reserved | PDATA | EEPROM | SYNC_CHANGE | DC_SYNC1 | DC_SYNC0 | DC_LATCH | AL_CONTROL |
| r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:24 | Reserved | Must be kept at reset value. |
| 23 | SYNC15_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 15 interrupt<br>1: SyncManager 15 interrupt pending |
| 22 | SYNC14_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 14 interrupt<br>1: SyncManager 14 interrupt pending |
| 21 | SYNC13_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 13 interrupt<br>1: SyncManager 13 interrupt pending |
| 20 | SYNC12_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 12 interrupt<br>1: SyncManager 12 interrupt pending |
| 19 | SYNC11_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 11 interrupt<br>1: SyncManager 11 interrupt pending |
| 18 | SYNC10_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 10 interrupt<br>1: SyncManager 10 interrupt pending |
| 17 | SYNC9_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 9 interrupt<br>1: SyncManager 9 interrupt pending |
| 16 | SYNC8_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 8 interrupt<br>1: SyncManager 8 interrupt pending |
| 15 | SYNC7_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 7 interrupt<br>1: SyncManager 7 interrupt pending |
| 14 | SYNC6_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 6 interrupt |

| | | |
|---|---|---|
| | | 1: SyncManager 6 interrupt pending |
| 13 | SYNC5_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 5 interrupt<br>1: SyncManager 5 interrupt pending |
| 12 | SYNC4_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 4 interrupt<br>1: SyncManager 4 interrupt pending |
| 11 | SYNC3_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 3 interrupt<br>1: SyncManager 3 interrupt pending |
| 10 | SYNC2_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 2 interrupt<br>1: SyncManager 2 interrupt pending |
| 9 | SYNC1_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 1 interrupt<br>1: SyncManager 1 interrupt pending |
| 8 | SYNC0_INT | SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]):<br>0: No SyncManager 0 interrupt<br>1: SyncManager 0 interrupt pending |
| 7 | Reserved | Must be kept at reset value. |
| 6 | PDATA | Watchdog Process Data:<br>0: Has not expired<br>1: Has expired<br>(Bit is cleared by reading Watchdog Status Process Data 0x0440 from PDI) |
| 5 | EEPROM | EEPROM Emulation:<br>0: No command pending<br>1: EEPROM command pending<br>(Bit is cleared by acknowledging the command in EEPROM Control/Status register 0x0502:0x0503[10:8] from PDI) |
| 4 | SYNC_CHANGE | SyncManager activation register (SyncManager register offset 0x6) changed:<br>0: No change in any SyncManager<br>1: At least one SyncManager changed<br>(Bit is cleared by reading SyncManager Activation registers 0x0806 etc. from PDI) |
| 3 | DC_SYNC1 | State of DC SYNC1 (if register 0x0151[7]=1):<br>(Bit is cleared by reading of SYNC1 status 0x098F from PDI, use only in Acknowledge mode) |
| 2 | DC_SYNC0 | State of DC SYNC0 (if register 0x0151[3]=1):<br>(Bit is cleared by reading SYNC0 status 0x098E from PDI, use only in Acknowledge |

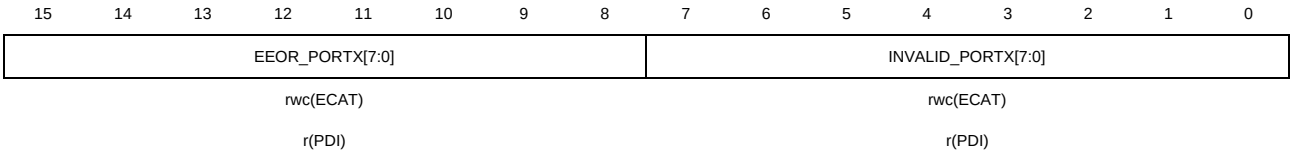
| | | |
|---|---|---|
| | | mode) |
| 1 | DC_LATCH | DC Latch event:<br>0: No change on DC Latch Inputs<br>1: At least one change on DC Latch Inputs<br>(Bit is cleared by reading DC Latch event times from PDI, so that Latch 0/1 Status 0x09AE:0x09AF indicates no event. Available if Latch Unit is PDI-controlled) |
| 0 | AL_CONTROL | AL Control event:<br>0: No AL Control Register change<br>1: AL Control Register has been written(AL control event is only generated if PDI emulation is turned off (ESC Configuration 0 register 0x0141[0]=0))<br>(Bit is cleared by reading AL Control register 0x0120:0x0121 from PDI) |

### RX Error Port X register (RX_PORTX_ERR) (X = 0,1,2,3)

Address Offset: 0x0300 + X * 2

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:8 | EEOR_PORTX[7:0] | Port X RX Error Counter<br>RX Error counter of Port X (counting is stopped when 0xFF is reached). |
| 7:0 | INVALID_PORTX[7:0] | Port X Invalid Frame Counter<br>Invalid frame counter of Port X (counting is stopped when 0xFF is reached). |

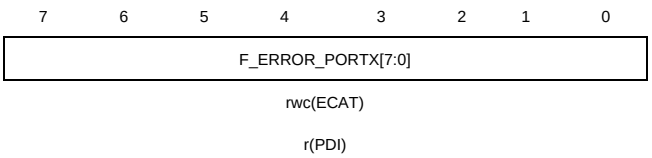
**NOTE:** Error Counters 0x0300-0x030B are cleared if one of the implemented RX Error counters 0x0300-0x030B is written (preferably 0x0300). Write value is ignored (write 0). Errors are only counted if the loop of the port is open.

### Forwarded RX Error Port X register (FWD_PORTX_ERR) (X = 0,1,2,3)

Address Offset: 0x0308 + X

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|

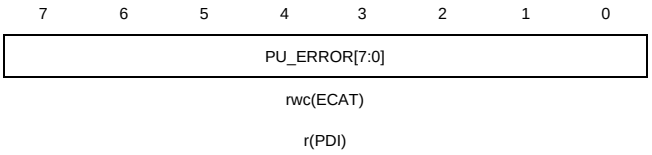
| | | |
|---|---|---|
| 7:0 | F_ERROR_PORTX[7:0] | Forwarded error counter of Port X (counting is stopped when 0xFF is reached). |

**NOTE:** Error Counters 0x0300-0x030B are cleared if one of the implemented RX Error counters 0x0300-0x030B is written (preferably 0x0300). Write value is ignored (write 0). Errors are only counted if the loop of the port is open.

### ESC Processing Unit Error register (ESC_PU_ERR)

Address Offset: 0x030C
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



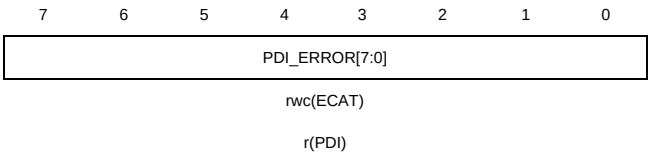

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | PU_ERROR[7:0] | ECAT Processing Unit error counter (counting is stopped when 0xFF is reached). Counts errors of frames passing the Processing Unit. |

**NOTE:** Error Counter 0x030C is cleared if error counter 0x030C is written. Write value is ignored (write 0).

### ESC PDI Error Counter register (ESC_PDI_ERR)

Address Offset: 0x030D
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



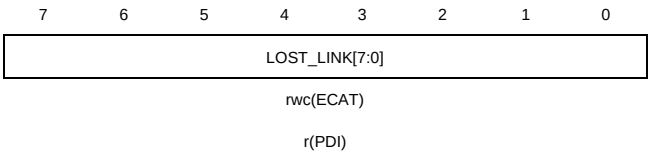

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | PDI_ERROR[7:0] | PDI Error counter (counting is stopped when 0xFF is reached). Counts if a PDI access has an interface error. |

**NOTE:** Error Counter 0x030D and Error Code 0x030E:0x030F are cleared if error counter 0x030D is written. Write value is ignored (write 0).

### ESC PORT X Lost Link register (ESC_PORTX_LOST_LINK) (X = 0,1,2,3)

Address Offset: 0x0310 + X
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



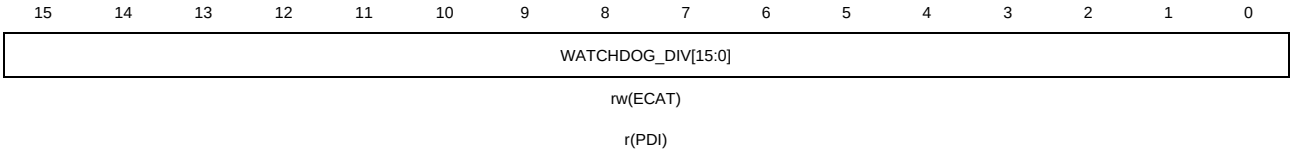

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | LOST_LINK[7:0] | Lost Link counter of Port X (counting is stopped when 0xff is reached). Counts only if port is open and loop is Auto. |

**NOTE:** Lost Link Counters 0x0310-0x0313 are cleared if one of the implemented Lost Link Counters 0x0310-0x0313 is written (preferably 0x0310). Write value is ignored (write 0).

### ESC Watchdog Divider register (ESC_WDG_DIV)

Address Offset: 0x0400
Reset value: 0x09C2

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

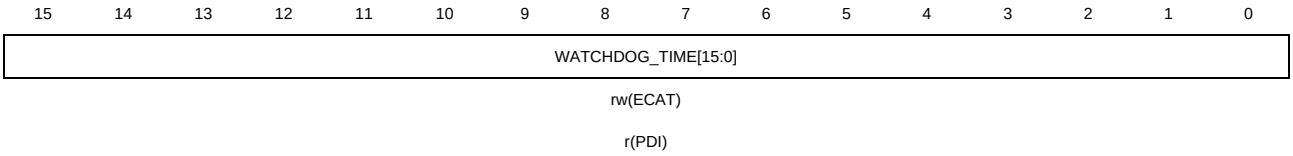



| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | WATCHDOG_DIV[15:0] | Watchdog divider: Number of 25 MHz tics (minus 2) that represent the basic watchdog increment. (Default value is 100µs = 2498) |

### ESC Watchdog Time PDI register (ESC_WDG_TIME_PDI)

Address Offset: 0x0410
Reset value: 0x03E8

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | WATCHDOG_TIME[15:0] | Watchdog Time PDI: number of basic watchdog increments (Default value with Watchdog divider 100µs means 100ms Watchdog) |

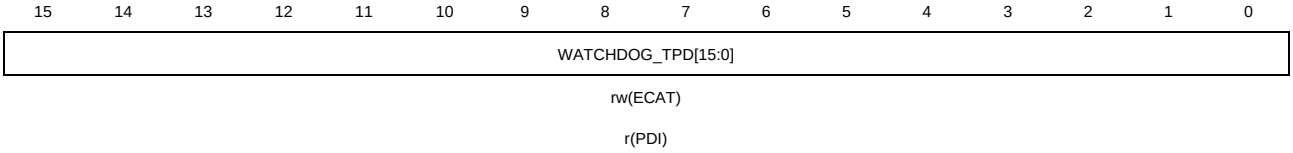
**NOTE:** Watchdog is disabled if Watchdog time is set to 0x0000. Watchdog starts counting again with every PDI access.

### ESC Watchdog Time Process Data register (ESC_WDG_TIME_PDATA)

Address Offset: 0x0420

Reset value: 0x03E8

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



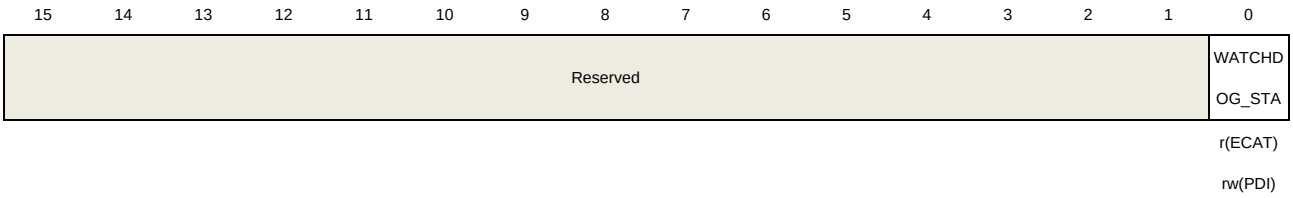

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | WATCHDOG_TPD[15:0] | Watchdog Time Process Data: number of basic watchdog increments (Default value with Watchdog divider 100µs means 100ms Watchdog) |

### ESC Watchdog Status Process Data register (ESC_WDG_STAT_PDATA)

Address Offset: 0x0440

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



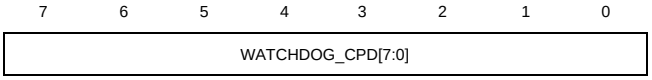

| Bits | Fields | Descriptions |
|---|---|---|
| 15:1 | Reserved | Must be kept at reset value. |
| 0 | WATCHDOG_STA | Watchdog Status of Process Data (triggered by SyncManagers)<br>0: Watchdog Process Data expired<br>1: Watchdog Process Data is active or disabled<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI clears AL Event Request 0x0220[6]. Writing to this register from PDI is not possible. PDI register function acknowledge by Write command is enabled: Writing this register from PDI clears AL Event Request 0x0220[6]. Writing to this register from PDI is possible; write value is ignored (write 0). |

### ESC Watchdog Counter Process Data register (ESC_WDG_CNT_PDATA)

Address Offset: 0x0442

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



rwc(ECAT)

r(PDI)

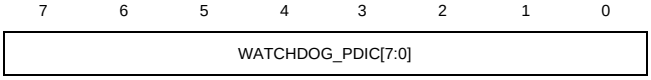
| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | WATCHDOG_CPD[7:0] | Watchdog Counter Process Data (counting is stopped when 0xFF is reached). Counts if Process Data Watchdog expires. |

### ESC Watchdog Counter PDI register (ESC_WDG_CNT_PDI)

Address Offset: 0x0443

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




rwc(ECAT)

r(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| 7:0 | WATCHDOG_PDIC[7:0] | Watchdog PDI counter (counting is stopped when 0xFF is reached). Counts if PDI Watchdog expires. |

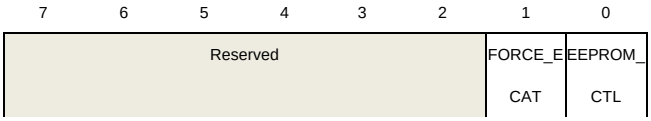
**NOTE:** Watchdog Counters 0x0442-0x0443 are cleared if one of the Watchdog Counters 0x0442-0x0443 is written. Write value is ignored (write 0).

### ESC EEPOM Configuration register (ESC_EEPROM_CFG)

Address Offset: 0x0500

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




rw(ECAT) rw(ECAT)

r(PDI) r(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | FORCE_ECAT | Force ECAT access:<br>0: Do not change Bit 0x0501[0]<br>1: Reset Bit 0x0501[0] to 0 |
| 0 | EEPROM_CTL | EEPROM control is offered to PDI:<br>0: no |

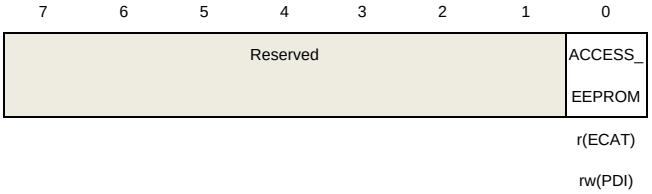
1: yes (PDI has EEPROM control)

### ESC EEPOM PDI Access register (ESC_EEPROM_ACC)

Address Offset: 0x0501

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | ACCESS_EEPROM | Access to EEPROM:<br>0: PDI releases EEPROM access<br>1: PDI takes EEPROM access (PDI has EEPROM control) |

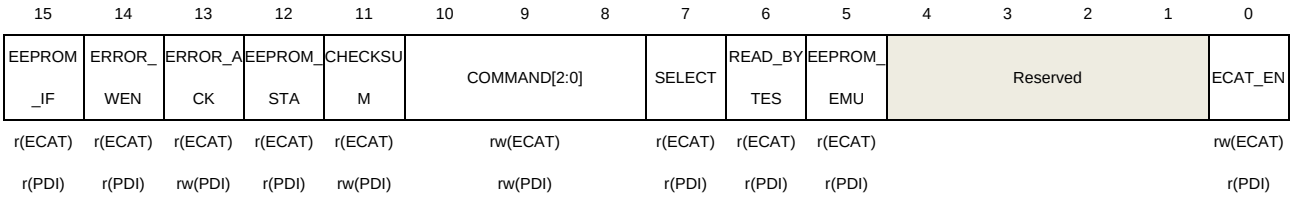
**NOTE:** write access is only possible if (0x0500[0]=1 or 0x0501[0]=1) and 0x0500[1]=0.

### ESC EEPOM Contorl/Status register (ESC_EEPROM_CS)

Address Offset: 0x0502

Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15 | EEPROM_IF | Busy(default 0):<br>0: EEPROM Interface is idle<br>1: EEPROM Interface is busy |
| 14 | ERROR_WEN | Error Write Enable(default 0):<br>0: No error<br>1: Write Command without Write enable<br>NOTE: Error bits are cleared by writing “000” (or any valid command) to Command Register Bits [10:8]. |
| 13 | ERROR_ACK | Error Acknowledge/Command(default 0): |

| | | |
|---|---|---|
| | | 0: No error<br>1: Missing EEPROM acknowledge or invalid command<br>EEPROM emulation only: PDI writes 1 if a temporary failure has occurred.<br>NOTE: Error bits are cleared by writing “000” (or any valid command) to Command Register Bits [10:8]. |
| 12 | EEPROM_STA | EEPROM loading status(default 0):<br>0: EEPROM loaded, device information ok<br>1: EEPROM not loaded, device information not available (EEPROM loading in progress or finished with a failure) |
| 11 | CHECKSUM | Checksum Error in ESC Configuration Area(default 0):<br>0: Checksum ok<br>1: Checksum error<br>EEPROM emulation for ESC only: PDI writes 1 if a CRC failure has occurred for a reload command. |
| 10:8 | COMMAND[2:0] | Command register(default 0):<br>Write: Initiate command.<br>Read: Currently executed command<br>Commands:<br>000: No command/EEPROM idle (clear error bits)<br>001: Read<br>010: Write<br>100: Reload<br>Others: Reserved/invalid commands (do not issue)<br>EEPROM emulation only: after execution, PDI writes command value to indicate operation is ready.<br>NOTE: Write Enable bit 0 is self-clearing at the SOF of the next frame, Command bits [10:8] are self-clearing after the command is executed (EEPROM Busy ends). Writing “000” to the command register will also clear the error bits [14:13]. Command bits [10:8] are ignored if Error Acknowledge/Command is pending (bit 13). |
| 7 | SELECT | Selected EEPROM Algorithm:<br>0: 1 address byte (1Kbit – 16Kbit EEPROMs)<br>1: 2 address bytes (32Kbit – 4 Mbit EEPROMs) |
| 6 | READ_BYTES | Supported number of EEPROM read bytes(default 0):<br>0: 4 Bytes<br>1: 8 Bytes |
| 5 | EEPROM_EMU | EEPROM emulation:<br>0: Normal operation (I²C interface used)<br>1: PDI emulates EEPROM (I²C not used) |
| 4:1 | Reserved | Must be kept at reset value. |
| 0 | ECAT_EN | ECAT write enable(default 0): |

0: Write requests are disabled

1: Write requests are enabled

This bit is always 1 if PDI has EEPROM control.

NOTE: Write Enable bit 0 is self-clearing at the SOF of the next frame, Command bits [10:8] are self-clearing after the command is executed (EEPROM Busy ends). Writing “000” to the command register will also clear the error bits [14:13]. Command bits [10:8] are ignored if Error Acknowledge/Command is pending (bit 13).

### ESC EEPROM Address register (ESC_EEPROM_ADDR)

Address Offset: 0x0504

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EEPROM_ADDR[31:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEPROM_ADDR[31:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | EEPROM_ADDR[31:0] | EEPROM Address<br>0: First word (= 16 bit)<br>1: Second word …<br>Actually used EEPROM Address bits:<br>[9:0]: EEPROM size up to 16 Kbit<br>[17:0]: EEPROM size 32 Kbit – 4 Mbit<br>[31:0]: EEPROM Emulation |

**NOTE:** write access depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is blocked if EEPROM interface is busy (0x0502[15]=1).

### ESC EEPROM Data register (ESC_EEPROM_DATA)

Address Offset: 0x0508

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EEPROM_READ[15:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| EERPOM_WRITE[15:0] |
|---|
| rw(ECAT) |
| rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:16 | EEPROM_READ[15:0] | EEPROM Read data (data read from EEPROM, higher bytes) |
| 15:0 | EERPOM_WRITE[15:0] | EEPROM Write data (data to be written to EEPROM) or EEPROM Read data (data read from EEPROM, lower bytes) |

### ESC MII Management Control / Status register (ESC_MII_MNG_CS)

Address Offset: 0x0510

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MII_IF | COMMAND_ERROR | READ_EEOR | Reserved | | | COMMAND[1:0] | | PHY_ADDRP0[4:0] | | | | | MII_CONFIG | PDI_CTL | WRITE_EN |
| r(ECAT) r(PDI) | r(ECAT) r(PDI) | rw(ECAT) rw(PDI) | | | | rw(ECAT) rw(PDI) | | r(ECAT) r(PDI) | | | | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | rw(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | MII_IF | Busy:<br>0: MII Management Interface is idle<br>1: MII Management Interface is busy |
| 14 | COMMAND_ERROR | Command error:<br>0: Last Command was successful<br>1: Invalid command or write command without Write Enable<br>Cleared by executing a valid command or by writing “00” to Command register bits [9:8]. |
| 13 | READ_EEOR | Read error:<br>0: No read error<br>1: Read error occurred (PHY or register not available)<br>Cleared by writing to register 0x0511. |
| 12:10 | Reserved | Must be kept at reset value. |
| 8:9 | COMMAND[1:0] | Command register:<br>Write: Initiate command.<br>Read: Currently executed command<br>00: No command/MI idle (clear error bits)<br>01: Read<br>10: Write<br>Others: Reserved/invalid command (do not issue)<br>NOTE: Write enable bit 0 is self-clearing at the SOF of the next frame, Command |

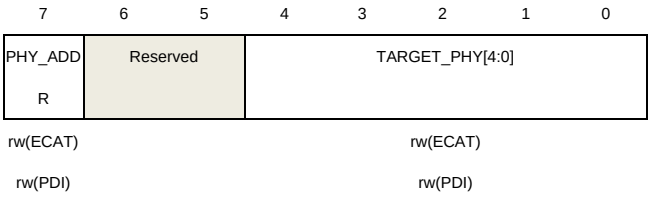
| | | |
|---|---|---|
| | | bits [10:8] are self-clearing after the command is executed (Busy ends). Writing “00” to the command register will also clear the error bits [14:13]. The Command bits are cleared after the command is executed. |
| 7:3 | PHY_ADDRP0[4:0] | PHY address of port 0<br>(this is equal to the PHY address offset, if the PHY addresses are consecutive)<br>IP Core since V3.0.0/3.00c: Translation 0x0512[7] =0:<br>Register 0x0510[7:3] shows PHY address of port 0<br>Translation 0x0512[7] =1: Register 0x0510[7:3] shows the PHY address which will be used for port 0-3 as requested by 0x0512[4:0] (valid values 0-3) |
| 2 | MII_CONFIG | MI link detection and configuration:<br>0: Disabled for all ports<br>1: Enabled for at least one MII port, refer to PHY Port Status (0x0518 ff.) for details |
| 1 | PDI_CTL | Management Interface can be controlled by PDI (registers 0x0516-0x0517):<br>0: Only ECAT control<br>1: PDI control possible |
| 0 | WRITE_EN | Write enable:<br>0: Write disabled<br>1: Write enabled<br>This bit is always 1 if PDI has MI control.<br>NOTE: Write enable bit 0 is self-clearing at the SOF of the next frame, Command bits [10:8] are self-clearing after the command is executed (Busy ends). Writing “00” to the command register will also clear the error bits [14:13]. The Command bits are cleared after the command is executed. |

### ESC PHY Address register (ESC_PHY_ADDR)

Address Offset: 0x0512

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7 | PHY_ADDR | Target PHY Address translation:<br>0: Enabled<br>1: Disabled<br>Refer to 0x0512[4:0] and 0x0510[7:3] for details. |
| 6:5 | Reserved | Must be kept at reset value. |

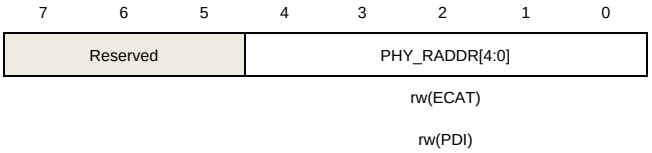
| | | |
|---|---|---|
| 4:0 | TARGET_PHY[4:0] | Target PHY Address<br>Translation 0x0512[7]=0:<br>0-3: Target PHY Addresses 0-3 are used to access the PHYs at port 0-3, when the PHY addresses are properly configured<br>4-31: The configured PHY address of port 0 (PHY address offset) is added to the Target PHY Address values 4-31 when accessing a PHY<br>Translation 0x0512[7]=1:<br>0-31: Target PHY Addresses is used when accessing a PHY without translation |

**NOTE:** write access depends on assignment of MI (ECAT/PDI). Write access is blocked if Management interface is busy (0x0510[15]=1).

### ESC PHY Register Address register (ESC_PHY_REG_ADDR)

Address Offset: 0x0513
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

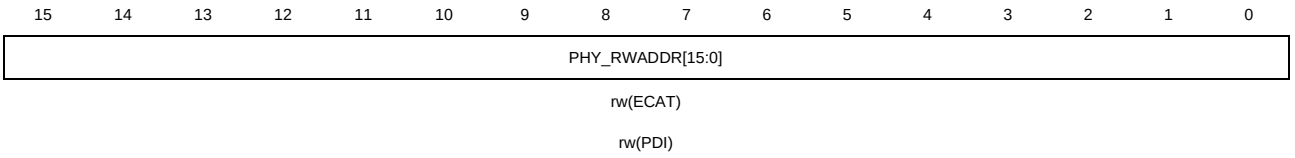



| Bits | Fields | Descriptions |
|---|---|---|
| 7:5 | Reserved | Must be kept at reset value. |
| 4:0 | PHY_RADDR[4:0] | Address of PHY Register that shall be read/written |

### ESC PHY Data register (ESC_PHY_DATA)

Address Offset: 0x0514
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

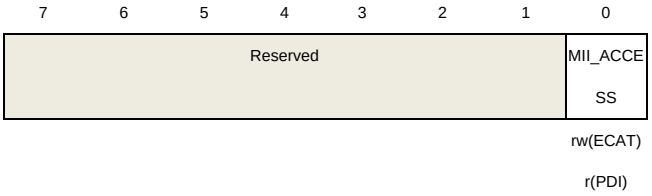



| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PHY_RWADDR[15:0] | PHY Read/Write Data |

### MII Management ECAT Access State register (MIIM_ECAT_ACC_STAT)

Address Offset: 0x0516
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | MII_ACCESS | Access to MII management:<br>0: ECAT enables PDI takeover of MII management interface<br>1: ECAT claims exclusive access to MII management interface |

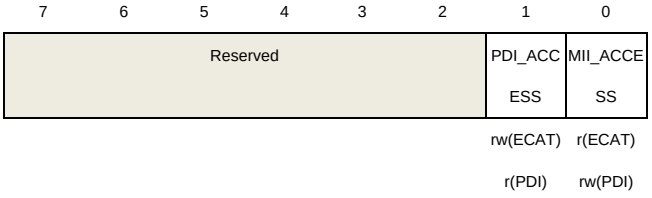
**NOTE:** write access is only possible if 0x0517[0]=0.

### MII Management ECAT Access State register (MIIM_PDI_ACC_STAT)

Address Offset: 0x0517

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | PDI_ACCESS | Force PDI Access State:<br>0: Do not change Bit 0x0517[0]<br>1: Reset Bit 0x0517[0] to 0 |
| 0 | MII_ACCESS | Access to MII management:<br>0: ECAT has access to MII management<br>1: PDI has access to MII management |

### PHY Port X Status register (PHY_PORTX_STAT) (X = 0,1,2,3)

Address Offset: 0x0518 + X

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| Reserved | PHY_UPDATE | PARTNER_ERROR | READ_ERROR | STATUS_ERROR | LINK_STATUS | PHYLINK |
|---|---|---|---|---|---|---|
| | rwc(ECAT) rwc(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | rwc(ECAT) rwc(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | PHY_UPDATE | PHY configuration updated:<br>0: No update<br>1: PHY configuration was updated<br>Cleared by writing any value to at least one of the PHY Port X Status registers. |
| 4 | PARTNER_ERROR | Link partner error:<br>0: No error detected<br>1: Link partner error |
| 3 | READ_ERROR | Read error:<br>0: No read error occurred<br>1: A read error has occurred<br>Cleared by writing any value to at least one of the PHY Port X Status registers. |
| 2 | STATUS_ERROR | Link status error:<br>0: No error<br>1: Link error, link inhibited |
| 1 | LINK_STATUS | Link status (100 Mbit/s, Full Duplex, Auto negotiation):<br>0: No link<br>1: Link detected |
| 0 | PHYLINK | Physical link status:<br>0: No physical link<br>1: Physical link detected |

### Logical Start address FMMU X register (FMMUX_LOGIC_ADDR) (X = 0…F)

Address Offset: 0x0600 + X * 0x10

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOGIC_ADDR[31:16] | | | | | | | | | | | | | | | |
| rw(ECAT) r(PDI) | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOGIC_ADDR[15:0] | | | | | | | | | | | | | | | |
| rw(ECAT) r(PDI) | | | | | | | | | | | | | | | |

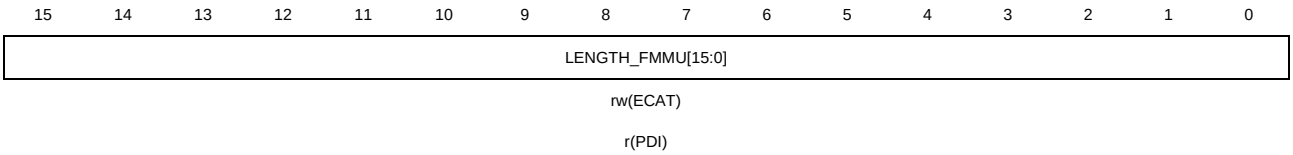
| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | LOGIC_ADDR[31:0] | Logical start address within the EtherCAT Address Space. |

### Length FMMU X register (FMMUX_LENGTH) (X = 0…F)

Address Offset: 0x0604 + X * 0x10

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



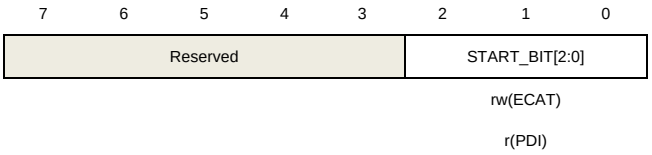

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | LENGTH_FMMU[15:0] | Offset from the first logical FMMU byte to the last FMMU byte + 1 (e.g., if two bytes are used, then this parameter shall contain 2) |

### Start bit FMMU X in Logical address space register (FMMUX_STRA_BIT) (X = 0…F)

Address Offset: 0x0606 + X * 0x10

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



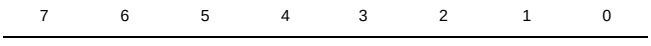

| Bits | Fields | Descriptions |
|---|---|---|
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | START_BIT[2:0] | Logical starting bit that shall be mapped (bits are counted from least significant bit 0 to most significant bit 7) |

### Stop bit FMMU X in Logical address space register (FMMUX_STOP_BIT) (X = 0…F)

Address Offset: 0x0607 + X * 0x10

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| Reserved | STOP_BIT[2:0] |
|---|---|
| | rw(ECAT) |
| | r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | STOP_BIT[2:0] | Last logical bit that shall be mapped (bits are counted from least significant bit 0 to most significant bit 7) |

### Physical Start address FMMU X register (FMMUX_ADDR) (X = 0…F)

Address Offset: 0x0608 + X * 0x10
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| START_ADDR[15:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | START_ADDR[15:0] | Physical Start Address (mapped to logical Start address) |

### Physical Start bit FMMU X register (FMMUX_PSBIT) (X = 0…F)

Address Offset: 0x060A + X * 0x10
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

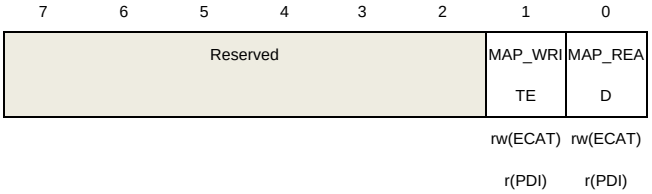
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | PSTART_BIT[2:0] | | |
| rw(ECAT) | | | | | | | |
| r(PDI) | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:3 | Reserved | Must be kept at reset value. |
| 2:0 | PSTART_BIT[2:0] | Physical starting bit as target of logical start bit mapping (bits are counted from least significant bit 0 to most significant bit 7) |

### Type FMMU X register (FMMUX_TYPE) (X = 0…F)

Address Offset: 0x060B + X * 0x10
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



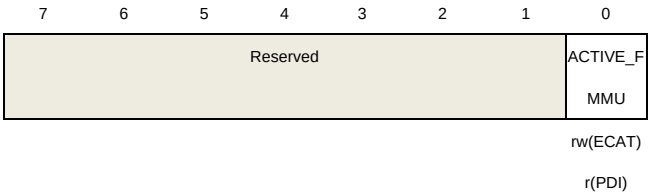

| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | MAP_WRITE | 0: Ignore mapping for write accesses<br>1: Use mapping for write accesses |
| 0 | MAP_READ | 0: Ignore mapping for read accesses<br>1: Use mapping for read accesses |

### Active FMMU X register (FMMUX_ACTIVE) (X = 0…F)

Address Offset: 0x060C + X * 0x10

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



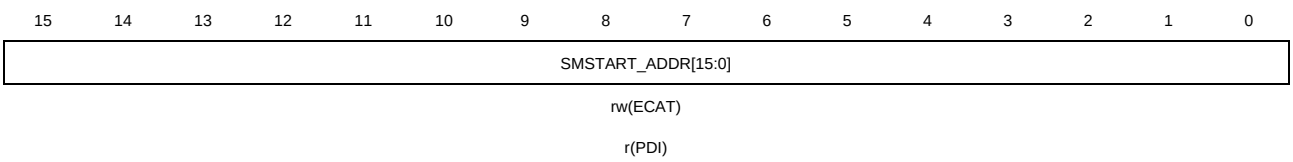

| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | ACTIVE_FMMU | 0: FMMU deactivated<br>1: FMMU activated. FMMU checks logically addressed blocks to be mapped according to configured mapping |

### Physical Start address SyncManager X register (SMX_START_ADDR) (X = 0…F)

Address Offset: 0x0800 + X * 8

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



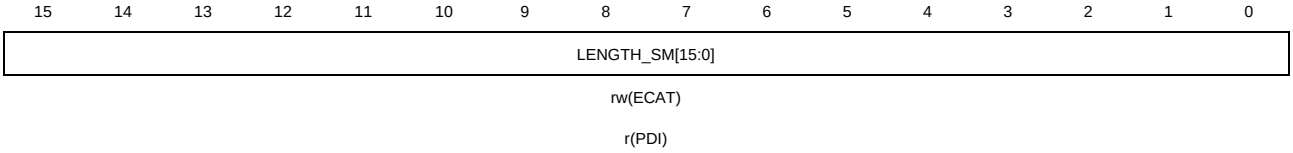

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | SMSTART_ADDR[15:0] | First byte that will be handled by SyncManager |

### Length SyncManager X register (SMX_LENGTH) (X = 0…F)

Address Offset: 0x0802 + X * 8

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



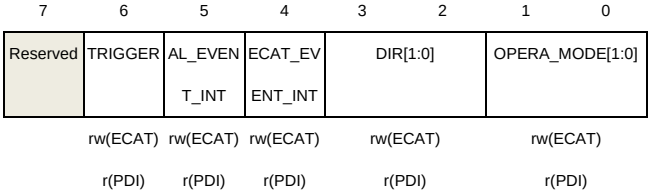

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | LENGTH_SM[15:0] | Number of bytes assigned to SyncManager (shall be greater than 1, otherwise SyncManager is not activated. If set to 1, only Watchdog Trigger is generated if configured) |

### Control Register SyncManager X register (SMX_CTL) (X = 0…F)

Address Offset: 0x0804 + X * 8

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7 | Reserved | Must be kept at reset value. |
| 6 | TRIGGER | Watchdog Trigger Enable:<br>0: Disabled<br>1: Enabled |
| 5 | AL_EVENT_INT | Interrupt in AL Event Request Register:<br>0: Disabled<br>1: Enabled |
| 4 | ECAT_EVENT_INT | Interrupt in ECAT Event Request Register:<br>0: Disabled<br>1: Enabled |
| 3:2 | DIR[1:0] | Direction:<br>00: Read: ECAT read access, PDI write access.<br>01: Write: ECAT write access, PDI read access.<br>10: Reserved |

| | | |
|---|---|---|
| | | 11: Reserved |
| 1:0 | OPERA_MODE[1:0] | Operation Mode:<br>00: Buffered (3 buffer mode)<br>01: Reserved<br>10: Mailbox (Single buffer mode)<br>11: Reserved |

### Status Register SyncManager X register (SMX_STAT) (X = 0…F)

Address Offset: 0x0805 + X * 8

Reset value: 0x30

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| WRITE_BUFFER | READ_BUFFER | BUFFER_STATUS[1:0] | | MAILBOX_STATUS | Reserved | READ_INT | WRITE_INT |
| r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) | | r(ECAT) r(PDI) | | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

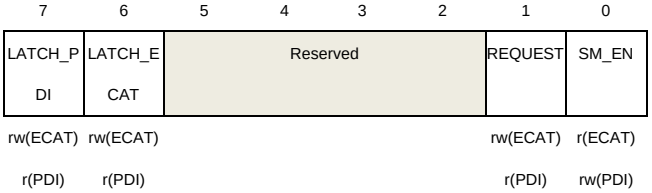
| Bits | Fields | Descriptions |
|---|---|---|
| 7 | WRITE_BUFFER | Write buffer in use (opened) |
| 6 | READ_BUFFER | Read buffer in use (opened) |
| 5:4 | BUFFER_STATUS[1:0] | Buffered mode: buffer status (last written buffer):<br>00: 1st buffer<br>01: 2nd buffer<br>10: 3rd buffer<br>11: (no buffer written)<br>Mailbox mode: reserved |
| 3 | MAILBOX_STATUS | Mailbox mode: mailbox status:<br>0: Mailbox empty<br>1: Mailbox full<br>Buffered mode: reserved |
| 2 | Reserved | Must be kept at reset value. |
| 1 | READ_INT | Interrupt Read:<br>1: Interrupt after buffer was completely and successfully read<br>0: Interrupt cleared after first byte of buffer was written<br>NOTE: This interrupt is signalled to the writing side if enabled in the SM Control register. |
| 0 | WRITE_INT | Interrupt Write:<br>1: Interrupt after buffer was completely and successfully written<br>0: Interrupt cleared after first byte of buffer was read |

NOTE: This interrupt is signalled to the reading side if enabled in the SM Control register.

### Activate SyncManager X register (SMX_ACTIVE) (X = 0…F)

Address Offset: 0x0806 + X * 8
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

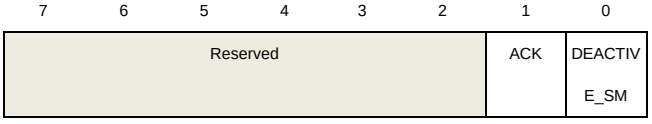



| Bits | Fields | Descriptions |
|---|---|---|
| 7 | LATCH_PDI | Latch Event PDI:<br>0: No<br>1: Generate Latch events when PDI issues a buffer exchange or when PDI accesses buffer start address |
| 6 | LATCH_ECAT | Latch Event ECAT:<br>0: No<br>1: Generate Latch event when EtherCAT main issues a buffer exchange |
| 5:2 | Reserved | Must be kept at reset value. |
| 1 | REQUEST | Repeat Request:<br>A toggle of Repeat Request means that a mailbox retry is needed (primarily used in conjunction with ECAT Read Mailbox) |
| 0 | SM_EN | SyncManager Enable/Disable:<br>0: Disable: Access to Memory without SyncManager control<br>1: Enable: SyncManager is active and controls Memory area set in configuration<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI in all SyncManagers which have changed activation clears AL Event Request 0x0220[4]. Writing to this register from PDI is not possible. |

### PDI Control SyncManager X register (SMX_PDI_CTL) (X = 0…F)

Address Offset: 0x0807 + X * 8
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | r(ECAT) rw(PDI) | r(ECAT) rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | ACK | Repeat Ack:<br>If this is set to the same value as that set by Repeat Request, the PDI acknowledges the execution of a previous set Repeat request. |
| 0 | DEACTIVE_SM | Deactivate SyncManager:<br>Read:<br>0: Normal operation, SyncManager activated.<br>1: SyncManager deactivated and reset. SyncManager locks access to Memory area.<br>Write:<br>0: Activate SyncManager<br>1: Request SyncManager deactivation<br>NOTE: Writing 1 is delayed until the end of the frame, which is currently processed. |

### ESC Receive Time Port 0 register (ESC_RCV_TIMEP0)

Address Offset: 0x0900

Reset value: Undefined

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCAL_TIMEP0[23:0] | | | | | | | | | | | | | | | |
| r(ECAT) r(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | | | | RWLOCAL_TIME[7:0] | | | | | | | |
| | | | | | | | | rw(ECAT) r(PDI) | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:8 | LOCAL_TIMEP0[23:0] | Local time at the beginning of the last receive frame containing a write access to register 0x0900. |
| 7:0 | RWLOCAL_TIME[7:0] | Write: A write access to register 0x0900 with BWR or FPWR latches the local time at the beginning of the receive frame (start first bit of preamble) at each port.<br>Read: Local time at the beginning of the last receive frame containing a write access to this register.<br>NOTE: FPWR requires an address match for accessing this register like any FPWR command. All write commands with address match will increment the working counter (e.g., APWR), but they will not trigger receive time latching. |

**NOTE:** The time stamps cannot be read in the same frame in which this register was written.

### ESC Receive Time Port 1 register (ESC_RCV_TIMEP1)

Address Offset: 0x0904

Reset value: Undefined

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCAL_TIMEP1[31:16] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LOCAL_TIMEP1[15:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | LOCAL_TIMEP1[31:0] | Local time at the beginning of a frame (start first bit of preamble) received at port 1 containing a BWR or FPWR to register 0x0900. |

### ESC Receive Time Port 2 register (ESC_RCV_TIMEP2)

Address Offset: 0x0908

Reset value: Undefined

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LOCAL_TIMEP2[31:16] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LOCAL_TIMEP2[15:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | LOCAL_TIMEP2[31:0] | Local time at the beginning of a frame (start first bit of preamble) received at port 2 containing a BWR or FPWR to register 0x0900. |

### ESC System Time register (ESC_SYS_TIME)

Address Offset: 0x0910

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. | 31 .. .. .. .. .. .. 0 |
|---|---|
| READ_ACESS[31:0] | WRITE_ACESS[31:0] |
| r(ECAT) r(PDI) | w(ECAT) w(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | READ_ACESS[63:0] | ECAT read access:<br>Local copy of the System Time when the frame passed the reference clock (i.e., including System Time Delay). Time latched at beginning of the frame (Ethernet SOF delimiter)<br>PDI read access:<br>Local copy of the System Time. Time latched when reading first byte (0x0910) |
| 31:0 | WRITE_ACESS[31:0] | Write access:<br>Written value will be compared with the local copy of the System time. The result is an input to the time control loop.<br>NOTE: written value will be compared at the end of the frame with the latched (SOF) local copy of the System time if at least the first byte (0x0910) was written. |

**NOTE:** Write access to this register depends upon ESC configuration (System Time PDI-controlled off=ECAT/ on=PDI; ECAT control is common).

### ESC Receive Time ECAT Processing Unit register (ESC_RCVT_ECAT_PU)

Address Offset: 0x0918
Reset value: Undefined

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. | 31 .. .. .. .. .. .. 0 |
|---|---|
| RECVE_TIME[63:0] | |
| r(ECAT) r(PDI) | |

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | RECVE_TIME[63:0] | Local time at the beginning of a frame (start first bit of preamble) received at the ECAT Processing Unit containing a write access to register 0x0900<br>NOTE: if port 0 is open, this register reflects the Receive Time Port 0 as a 64 Bit value. Any valid EtherCAT write access to register 0x0900 triggers latching, not only BWR/FPWR commands as with register 0x0900. |

### ESC System Time Offset register (ESC_SYS_TIME_OFFSET)

Address Offset: 0x0920
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. | 31 .. .. .. .. .. .. 0 |
|---|---|

| OFFSET_TIME[63:0] |
|---|
| rw(ECAT) |
| rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | OFFSET_TIME[63:0] | Difference between local time and System Time. Offset is added to the local time. |

### ESC System Time Delay register (ESC_SYS_TIME_DELAY)

Address Offset: 0x0928

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DELAY_TIME[31:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DELAY_TIME[31:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | DELAY_TIME[31:0] | Delay between Reference Clock and the ESC |

### ESC System Time Difference register (ESC_SYS_TIME_DIFF)

Address Offset: 0x092C

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIME_COMPARE | DIFF_TIME[30:0] | | | | | | | | | | | | | | |
| r(ECAT) | r(ECAT) | | | | | | | | | | | | | | |
| r(PDI) | r(PDI) | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DIFF_TIME[30:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31 | TIME_COMPARE | 0: Local copy of System Time less than received System Time<br>1: Local copy of System Time greater than or equal to received System Time |

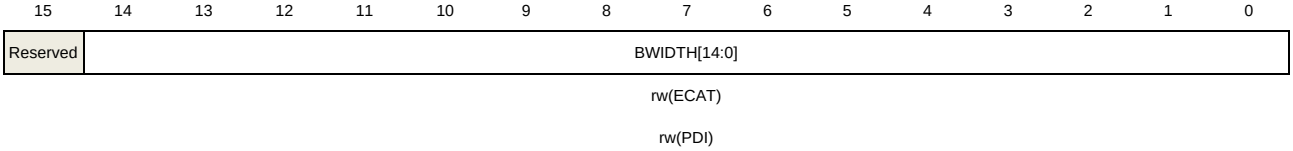
| | | |
|---|---|---|
| 30:0 | DIFF_TIME[30:0] | Mean difference between local copy of System Time and received System Time values<br>Difference = Received System Time – local copy of System Time |

**NOTE:** Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

### ESC Speed Counter Start register (ESC_SPD_CNT_START)

Address Offset: 0x0930
Reset value: 0x1000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



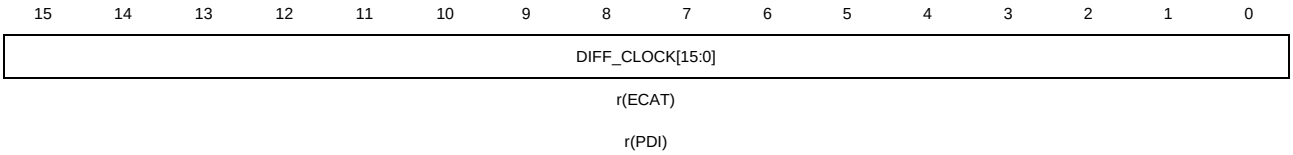

| Bits | Fields | Descriptions |
|---|---|---|
| 15 | Reserved | Must be kept at reset value. |
| 14:0 | BWIDTH[14:0] | Bandwidth for adjustment of local copy of System Time (larger values -> smaller bandwidth and smoother adjustment)<br>A write access resets System Time Difference (0x092C:0x092F) and Speed Counter Diff (0x0932:0x0933). Valid values: 0x0080 to 0x3FFF |

**NOTE:** Write access to this register depends upon ESC configuration (System Time PDI-controlled off=ECAT / on=PDI; ECAT control is common).

### ESC Speed Counter Diff register (ESC_SPD_CNT_DIFF)

Address Offset: 0x0932
Reset value: 0x0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | DIFF_CLOCK[15:0] | Representation of the deviation between local clock period and Reference Clock's clock period (representation: two's complement)<br>Range: ± (Speed Counter Start – 0x7F) |

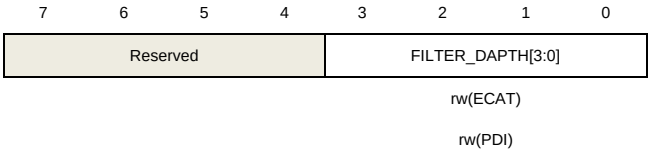
**NOTE:** The clock deviation after System Time Difference has settled at a low value can be calculated as follows:

Deviation = Speed Counter Diff / 5(Speed Counter Start + Speed Counter Diff + 2)(Speed Counter Start - Speed Counter Diff + 2)

### ESC System Time Difference Filter Depth register (ESC_SYS_TIME_DIFF_FD)

Address Offset: 0x0934
Reset value: 0x04

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



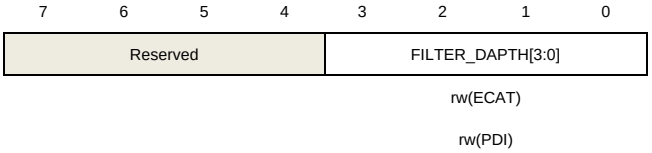

| Bits | Fields | Descriptions |
|---|---|---|
| 7:4 | Reserved | Must be kept at reset value. |
| 3:0 | FILTER_DAPTH[3:0] | Filter depth for averaging the received System Time deviation |

**NOTE:** Write access to this register depends upon ESC configuration (System Time PDI-controlled off=ECAT/ on=PDI; ECAT control is common).

### ESC Speed Counter Filter Depth register (ESC_SPD_CNT_FD)

Address Offset: 0x0935
Reset value: 0x12

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



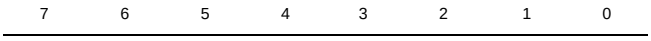

| Bits | Fields | Descriptions |
|---|---|---|
| 7:4 | Reserved | Must be kept at reset value. |
| 3:0 | FILTER_DAPTH[3:0] | Filter depth for averaging the clock period deviation |

**NOTE:** Write access to this register depends upon ESC configuration (System Time PDI-controlled off=ECAT/ on=PDI; ECAT control is common).

### ESC Cyclic Unit Control register (ESC_CYC_UNIT_CTL)

Address Offset: 0x0980
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



| Reserved | LATCH_UNIT1 | LATCH_UNIT0 | Reserved | UNIT_CTL |
|---|---|---|---|---|
| | rw(ECAT) r(PDI) | rw(ECAT) r(PDI) | | rw(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:6 | Reserved | Must be kept at reset value. |
| 5 | LATCH_UNIT1 | Latch In unit 1:<br>0: ECAT-controlled<br>1: PDI-controlled<br>NOTE: Latch interrupt is routed to ECAT/PDI depending on this setting |
| 4 | LATCH_UNIT0 | Latch In unit 0:<br>0: ECAT-controlled<br>1: PDI-controlled<br>NOTE: Latch interrupt is routed to ECAT/PDI depending on this setting. Always 1 (PDI-controlled) if System Time is PDI-controlled. |
| 3:1 | Reserved | Must be kept at reset value. |
| 0 | UNIT_CTL | Cyclic Unit and SYNC0 out unit control:<br>0: ECAT-controlled<br>1: PDI-controlled |

### ESC Register Activation register (ESC_REG_ACT)

Address Offset: 0x0981

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SYNCSIGNAL | CONFIG | START_TIME | EXTEN | AUTO_ACTIVE | SYNC1 | SYNC0 | SYNC_OUT |
| rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) | rw(ECAT) rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7 | SYNCSIGNAL | SyncSignal debug pulse (Vasily bit):<br>0: Deactivated<br>1: Immediately generate one ping only on SYNC0-1 according to 0x0981[2:1 for debugging<br>This bit is self-clearing, always read 0.<br>All pulses are generated at the same time, the cycle time is ignored. The configured pulse length is used. |
| 6 | CONFIG | Near future configuration (approx.): |

| | | |
|---|---|---|
| | | 0: ½ DC width future ($2^{31}$ ns or $2^{63}$ ns)<br>1: ~2.1 sec. future ($2^{31}$ ns) |
| 5 | START_TIME | Start Time plausibility check:<br>0: Disabled. SyncSignal generation if Start Time is reached.<br>1: Immediate SyncSignal generation if Start Time is outside near future (see 0x0981[6]) |
| 4 | EXTEN | Extension of Start Time Cyclic Operation (0x0990:0x0993):<br>0: No extension<br>1: Extend 32 bit written Start Time to 64 bit |
| 3 | AUTO_ACTIVE | Auto-activation by writing Start Time Cyclic Operation (0x0990:0x0997):<br>0: Disabled<br>1: Auto-activation enabled. 0x0981[0] is set automatically after Start Time is written. |
| 2 | SYNC1 | SYNC1 generation:<br>0: Deactivated<br>1: SYNC1 pulse is generated |
| 1 | SYNC0 | SYNC0 generation:<br>0: Deactivated<br>1: SYNC0 pulse is generated |
| 0 | SYNC_OUT | Sync Out Unit activation:<br>0: Deactivated<br>1: Activated |

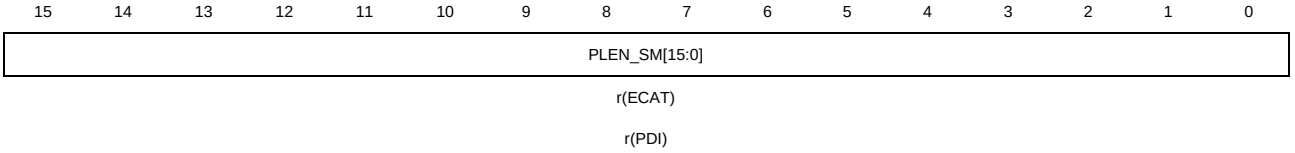
**NOTE:** Write to this register depends upon setting of 0x0980[0]

### ESC Pulse Length of SyncSignals register (ESC_PULSE_SM)

Address Offset: 0x0982

Reset value: depends on configuration

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



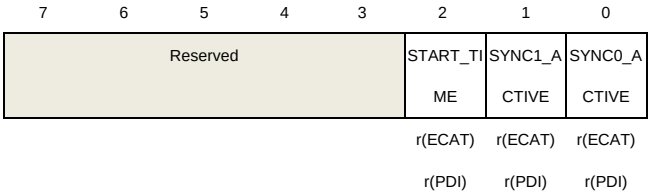

| Bits | Fields | Descriptions |
|---|---|---|
| 15:0 | PLEN_SM[15:0] | Pulse length of SyncSignals (in Units of 10ns)<br>0: Acknowledge mode: SyncSignal will be cleared by reading SYNC[1:0] Status register |

### ESC Activation Status register (ESC_ACT_STAT)

Address Offset: 0x0984

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



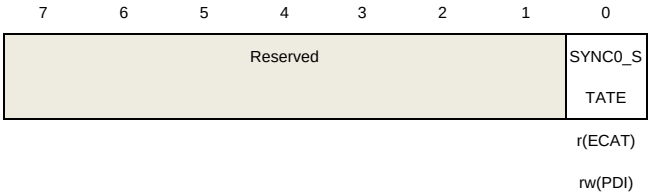

| Bits | Fields | Descriptions |
|---|---|---|
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | START_TIME | Start Time Cyclic Operation (0x0990:0x0997) plausibility check result when Sync Out Unit was activated:<br>0: Start Time was within near future<br>1: Start Time was out of near future (0x0981[6]) |
| 1 | SYNC1_ACTIVE | SYNC1 activation state:<br>0: First SYNC1 pulse is not pending<br>1: First SYNC1 pulse is pending |
| 0 | SYNC0_ACTIVE | SYNC0 activation state:<br>0: First SYNC0 pulse is not pending<br>1: First SYNC0 pulse is pending |

### ESC SYNC0 Status register (ESC_SYNC0_STAT)

Address Offset: 0x098E

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



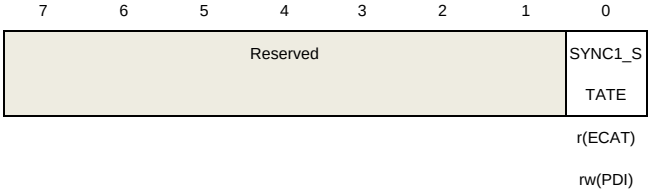

| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | SYNC0_STATE | SYNC0 state for Acknowledge mode.<br>SYNC0 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI clears AL Event Request 0x0220[2]. Writing to this register from PDI is not possible.<br>PDI register function acknowledge by Write command is enabled: Writing this |

register from PDI clears AL Event Request 0x0220[2]. Writing to this register from PDI is possible; write value is ignored (write 0).

### ESC SYNC1 Status register (ESC_SYNC1_STAT)

Address Offset: 0x098F
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

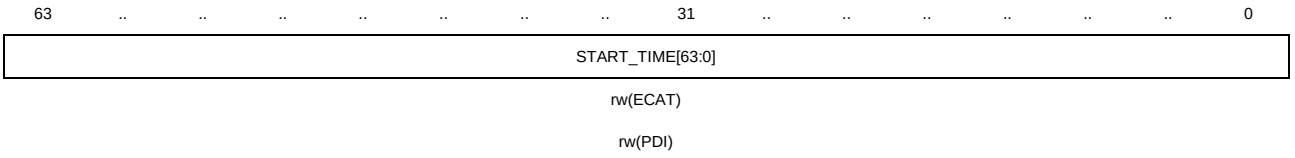



| Bits | Fields | Descriptions |
|---|---|---|
| 7:1 | Reserved | Must be kept at reset value. |
| 0 | SYNC1_STATE | SYNC1 state for Acknowledge mode.<br>SYNC1 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI clears AL Event Request 0x0220[3]. Writing to this register from PDI is not possible.<br>PDI register function acknowledge by Write command is enabled: Writing this register from PDI clears AL Event Request 0x0220[3]. Writing to this register from PDI is possible; write value is ignored (write 0). |

### ESC Start Time Cyclic Operation register (ESC_START_TIME_CO)

Address Offset: 0x0990
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | START_TIME[63:0] | Write: Start time (System time) of cyclic operation in ns<br>Read: System time of next SYNC0 pulse in ns |

**NOTE:** Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

### ESC Next SYNC1 Pulse register (ESC_NEXT_SYNC1_PLUSE)

Address Offset: 0x0998

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 | .. | .. | .. | .. | .. | .. | .. | 31 | .. | .. | .. | .. | .. | .. | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NEXT_SYNC1[63:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | NEXT_SYNC1[63:0] | System time of next SYNC1 pulse in ns |

**NOTE:** Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

### ESC SYNC0 Cycle Time register (ESC_SYNC0_CYC_TIME)

Address Offset: 0x09A0

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYNC0_CYCLE[31:16] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYNC0_CYCLE[15:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SYNC0_CYCLE[31:0] | Time between two consecutive SYNC0 pulses in ns.<br>0: Single shot mode, generate only one SYNC0 pulse.<br>NOTE: Write to this register depends upon setting of 0x0980[0]. Minimum value for cyclic operation: 60 [ns]. |

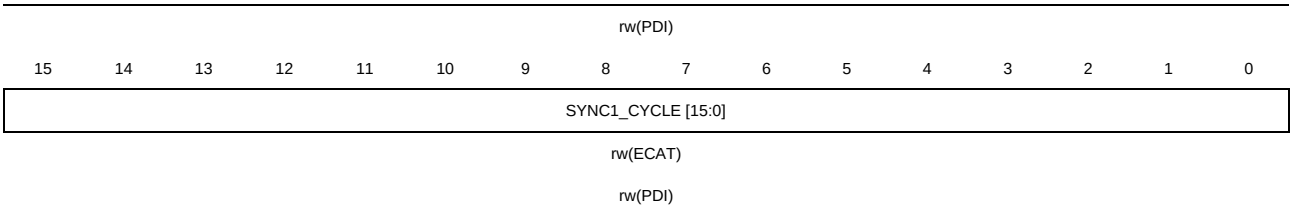
### ESC SYNC1 Cycle Time register (ESC_SYNC1_CYC_TIME)

Address Offset: 0x09A4

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SYNC1_CYCLE[31:16] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |



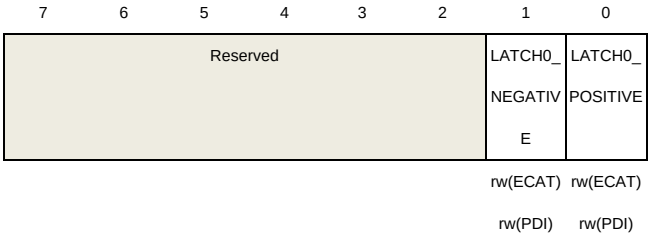

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | SYNC1_CYCLE[31:0] | Time between SYNC0 pulse and SYNC1 pulse in ns<br>NOTE: Write to this register depends upon setting of 0x0980[0]. |

### ESC Latch0 Control register (ESC_LATCH0_CTL)

Address Offset: 0x09A8

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | LATCH0_NEGATIVE | Latch0 negative edge:<br>0: Continuous Latch active<br>1: Single event (only first event active) |
| 0 | LATCH0_POSITIVE | Latch0 positive edge:<br>0: Continuous Latch active<br>1: Single event (only first event active) |

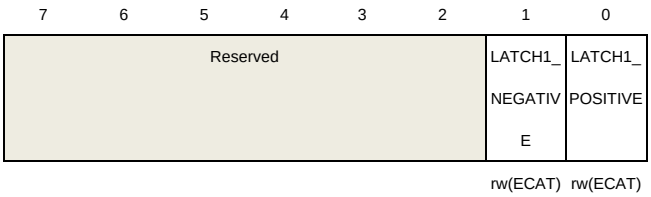
**NOTE:** Write access depends upon setting of 0x0980[4].

### ESC Latch1 Control register (ESC_LATCH1_CTL)

Address Offset: 0x09A9

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



rw(PDI) rw(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| 7:2 | Reserved | Must be kept at reset value. |
| 1 | LATCH1_NEGATIVE | Latch1 negative edge:<br>0: Continuous Latch active<br>1: Single event (only first event active) |
| 0 | LATCH1_POSITIVE | Latch1 positive edge:<br>0: Continuous Latch active<br>1: Single event (only first event active) |

**NOTE:** Write access depends upon setting of 0x0980[5].

### ESC Latch0 Status register (ESC_LATCH0_STAT)

Address Offset: 0x09AE
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

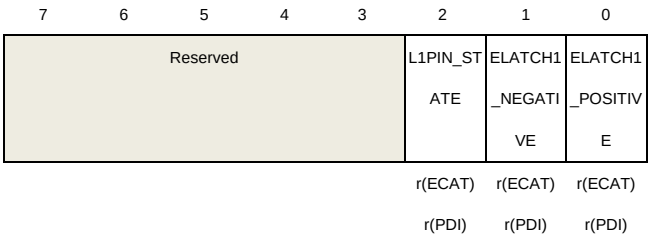
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | L0PIN_STATE | ELATCH0_NEGATIVE | ELATCH0_POSITIVE |
| | | | | | r(ECAT) r(PDI) | r(ECAT) r(PDI) | r(ECAT) r(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | L0PIN_STATE | Latch0 pin state |
| 1 | ELATCH0_NEGATIVE | Event Latch0 negative edge.<br>0: Negative edge not detected or continuous mode<br>1: Negative edge detected in single event mode only.<br>Flag cleared by reading out Latch0 Time Negative Edge. |
| 0 | ELATCH0_POSITIVE | Event Latch0 positive edge.<br>0: Positive edge not detected or continuous mode<br>1: Positive edge detected in single event mode only.<br>Flag cleared by reading out Latch0 Time Positive Edge. |

### ESC Latch1 Status register (ESC_LATCH1_STAT)

Address Offset: 0x09AF
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

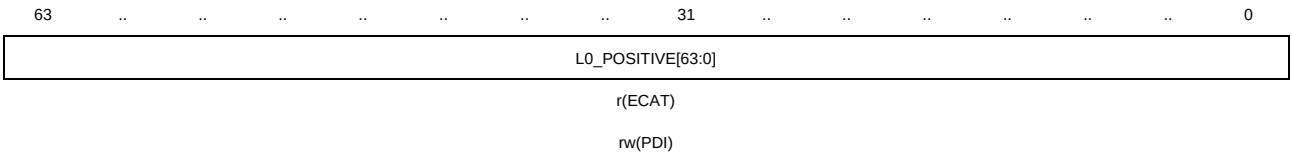



| Bits | Fields | Descriptions |
|---|---|---|
| 7:3 | Reserved | Must be kept at reset value. |
| 2 | L1PIN_STATE | Latch0 pin state |
| 1 | ELATCH1_NEGATIVE | Event Latch1 negative edge.<br>0: Negative edge not detected or continuous mode<br>1: Negative edge detected in single event mode only.<br>Flag cleared by reading out Latch1 Time Negative Edge. |
| 0 | ELATCH1_POSITIVE | Event Latch1 positive edge.<br>0: Positive edge not detected or continuous mode<br>1: Positive edge detected in single event mode only.<br>Flag cleared by reading out Latch1 Time Positive Edge. |

### ESC Latch0 Time Positive Edge (ESC_LATCH0_TIME_PE)

Address Offset: 0x09B0
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | L0_POSITIVE[63:0] | System time at the positive edge of the Latch0 signal.<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI if 0x0980[4]=1 clears Latch0 Status 0x09AE[0]. Writing to this register from PDI is not possible.<br>PDI register function acknowledge by Write command is enabled: Writing this register from PDI if 0x0980[4]=1 clears Latch0 Status 0x09AE[0]. Writing to this register from PDI is possible; write value is ignored (write 0). |

**NOTE:** Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch0 Status 0x09AE[0] if 0x0980[4]=0. Writing to this register from ECAT is not possible.

### ESC Latch0 Time Negative Edge (ESC_LATCH0_TIME_NE)

Address Offset: 0x09B8

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. 31 .. .. .. .. .. .. 0 |
|---|
| L0_NEGATIVE[63:0] |
| r(ECAT) |
| rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | L0_NEGATIVE[63:0] | System time at the negative edge of the Latch0 signal.<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI if 0x0980[4]=1 clears Latch0 Status 0x09AE[1]. Writing to this register from PDI is not possible.<br>PDI register function acknowledge by Write command is enabled: Writing this register from PDI if 0x0980[4]=1 clears Latch0 Status 0x09AE[1]. Writing to this register from PDI is possible; write value is ignored (write 0). |

**NOTE:** Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch0 Status 0x09AE[1] if 0x0980[4]=0. Writing to this register from ECAT is not possible.

### ESC Latch1 Time Positive Edge (ESC_LATCH1_TIME_PE)

Address Offset: 0x09C0

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. 31 .. .. .. .. .. .. 0 |
|---|
| L1_POSITIVE[63:0] |
| r(ECAT) |
| rw(PDI) |

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | L1_POSITIVE[63:0] | System time at the positive edge of the Latch1 signal.<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI if 0x0980[5]=1 clears Latch0 Status 0x09AF[0]. Writing to this register from PDI is not possible.<br>PDI register function acknowledge by Write command is enabled: Writing this register from PDI if 0x0980[5]=1 clears Latch0 Status 0x09AF[0]. Writing to this register from PDI is possible; write value is ignored (write 0). |

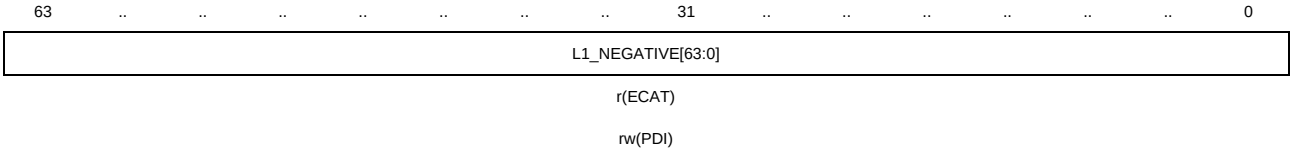
**NOTE:** Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0]

are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch0 Status 0x09AF[0] if 0x0980[5]=0. Writing to this register from ECAT is not possible.

### ESC Latch1 Time Negative Edge (ESC_LATCH1_TIME1_NE)

Address Offset: 0x09C8
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



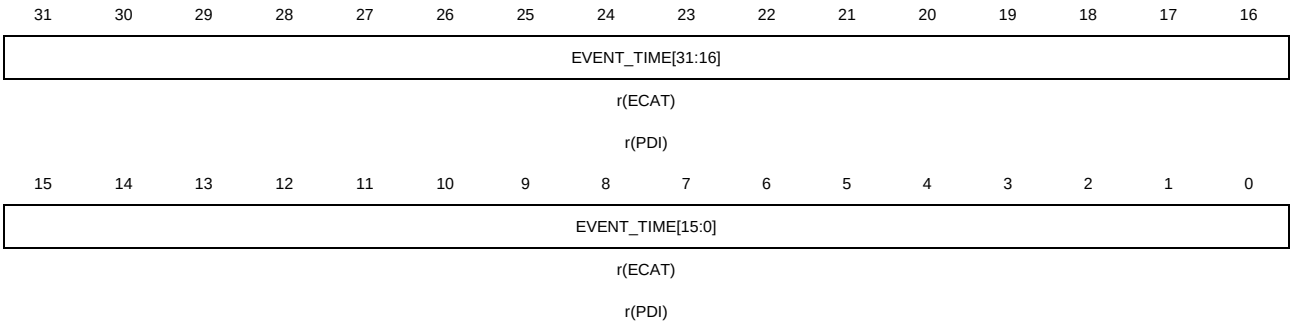

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | L1_NEGATIVE[63:0] | System time at the negative edge of the Latch1 signal.<br>NOTE: PDI register function acknowledge by Write command is disabled: Reading this register from PDI if 0x0980[5]=1 clears Latch0 Status 0x09AF[1]. Writing to this register from PDI is not possible.<br>PDI register function acknowledge by Write command is enabled: Writing this register from PDI if 0x0980[5]=1 clears Latch0 Status 0x09AF[1]. Writing to this register from PDI is possible; write value is ignored (write 0). |

**NOTE:** Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Reading this register from ECAT clears Latch0 Status 0x09AF[1] if 0x0980[5]=0. Writing to this register from ECAT is not possible.

### ESC Buffer Change Event Time register (ESC_BUF_CET)

Address Offset: 0x09F0
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)




| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | EVENT_TIME[31:0] | Local time at the beginning of the frame which causes at least one SyncManager |

to assert an ECAT event

**NOTE:** Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

### ESC PDI Buffer Start Event Time register (ESC_PDI_BUF_SET)

Address Offset: 0x09F8
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SEVENT_TIME[31:16] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SEVENT_TIME[15:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PDI_SEVENT_TIME[31:0] | Local time when at least one SyncManager asserts a PDI buffer start event |

**NOTE:** Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

### ESC PDI Buffer Change Event Time register (ESC_PDI_BUF_CET)

Address Offset: 0x09FC
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CEVENT_TIME[31:16] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CEVENT_TIME[15:0] | | | | | | | | | | | | | | | |
| r(ECAT) | | | | | | | | | | | | | | | |
| r(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | PDI_CEVENT_TIME[31:0] | Local time when at least one SyncManager asserts a PDI buffer change event |

**NOTE:** Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0]

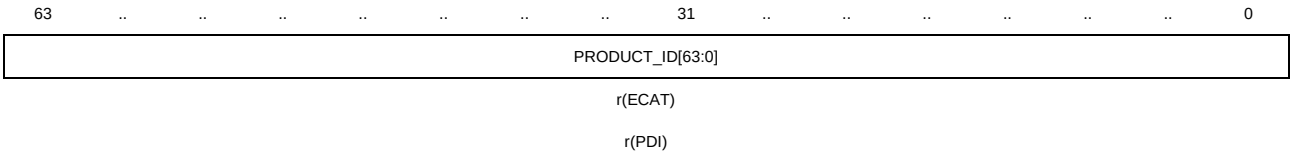
are read, which guarantees reading a consistent value.

### ESC Product ID register (ESC_PRODUCT_ID)

Address Offset: 0x0E00

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



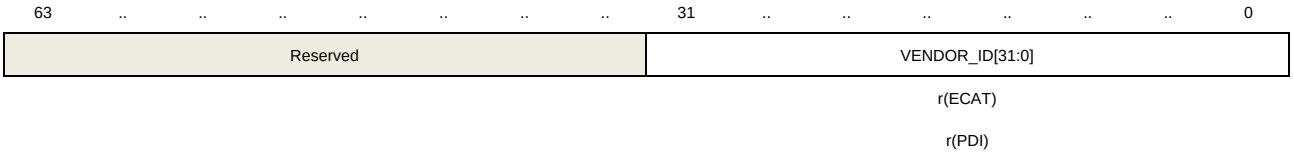

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | PRODUCT_ID[63:0] | Product ID |

### ESC Vendor ID register (ESC_VENDOR_ID)

Address Offset: 0x0E08

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



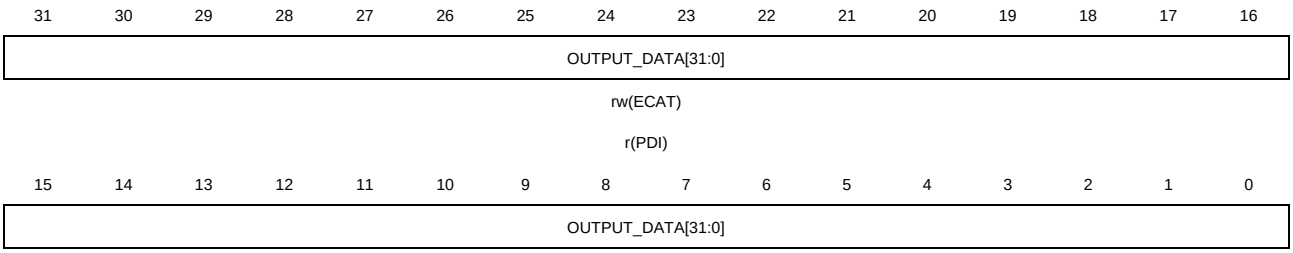

| Bits | Fields | Descriptions |
|---|---|---|
| 63:32 | Reserved | Must be kept at reset value. |
| 31:0 | VENDOR_ID[31:0] | Vendor ID:<br>[23:0] Company<br>[31:24] Department<br>NOTE: Test Vendor IDs have [31:28]=0xE |

### ESC Digital I/O Output Data register (ESC_DIG_DATA)

Address Offset: 0x0F00

Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



rw(ECAT)

r(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | OUTPUT_DATA[31:0] | Output Data |

**NOTE:** Register size depends on PDI setting and/or device configuration. This register is bit-writable (using Logical addressing).

### ESC General Purpose Outputs register (ESC_GP_OUTPUT)

Address Offset: 0x0F10
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. | 31 .. .. .. .. .. .. 0 |
|---|---|
| GP_OUTPUT[63:0] | |

rw(ECAT)

rw(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | GP_OUTPUT [63:0] | General Purpose Output Data |

**NOTE:** Register size depends on PDI setting and/or device configuration

### ESC General Purpose Inputs register (ESC_GP_INPUT)

Address Offset: 0x0F18
Reset value: 0x0

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 63 .. .. .. .. .. .. .. | 31 .. .. .. .. .. .. 0 |
|---|---|
| GP_INPUT[63:0] | |

r(ECAT)

r(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| 63:0 | GP_INPUT[63:0] | General Purpose Input Data |

**NOTE:** Register size depends on PDI setting and/or device configuration

### ESC User RAM register (ESC_USER_RAM)

Address Offset: 0x0F80

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 1023 .. .. .. .. .. .. .. .. .. .. .. .. .. .. 0 |
|---|
| USER_RAM [1023:0] |

rw(ECAT)

rw(PDI)

| Bits | Fields | Descriptions |
|---|---|---|
| ---- | USER_RAM [1023:0] | Application-specific information(128 Bytes) |

### ESC PDI Digital I/O Input Data register (ESC_PDI_DATA)

Address Offset: 0x1000
Reset value: undefined

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INPUT_DATA[31:16] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INPUT_DATA[15:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| 31:0 | INPUT_DATA[31:0] | Input Data |

**NOTE:** Process Data RAM is only accessible if EEPROM was correctly loaded (register 0x0110[0] = 1).

Input Data size depends on PDI setting and/or device configuration. Digital I/O Input Data is written into the Process Data RAM at these addresses if a Digital I/O PDI with inputs is configured.

### ESC Process Data RAM register (ESC_PDRAM)

Address Offset: 0x1000
Reset value: undefined

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

| 8192 | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PDRAM[8192:0] | | | | | | | | | | | | | | | |
| rw(ECAT) | | | | | | | | | | | | | | | |
| rw(PDI) | | | | | | | | | | | | | | | |

| Bits | Fields | Descriptions |
|---|---|---|
| ---- | PDRAM [8192:0] | Process Data RAM |

**NOTE:** Process Data RAM is only accessible if EEPROM was correctly loaded (register

0x0110[0] = 1).

# 51. Appendix

## 51.1. List of abbreviations used in register

**Table 51-1. List of abbreviations used in register**

| abbreviations for registers | Descriptions |
|---|---|
| read/write (rw) | Software can read and write to this bit. |
| read-only (r) | Software can only read this bit. |
| write-only (w) | Software can only write to this bit. Reading this bit returns the reset value. |
| read/clear write 1 (rc_w1) | Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value. |
| read/clear write 0 (rc_w0) | Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value. |
| toggle (t) | The software can toggle this bit by writing 1. Writing 0 has no effect. |
| read/set (rs) | Software can read as well as set this bit to 1. Writing '0' has no effect on the bit value. |
| read/clear by read (rc_r) | Software can read this bit. Reading this bit automatically clears it to '0'. Writing '0' has no effect on the bit value. |

## 51.2. List of terms

**Table 51-2. List of terms**

| Glossary | Descriptions |
|---|---|
| Word | Data of 32-bit length. |
| Half-word | Data of 16-bit length. |
| Byte | Data of 8-bit length. |
| IAP (in-application programming) | Writing 0 has no effect IAP is the ability to re-program the Flash memory of a microcontroller while the user program is running. |
| ICP (in-circuit programming) | ICP is the ability to program the Flash memory of a microcontroller using the JTAG protocol, the SWD protocol or the boot loader while the device is mounted on the user application board. |
| Option bytes | Product configuration bits stored in the Flash memory. |
| AHB | Advanced high-performance bus. |
| APB | Advanced peripheral bus. |
| RAZ | Read-as-zero. |
| WI | Writes ignored. |
| RAZ/WI | Read-as-zero, writes ignored. |

## 51.3. Available peripherals

For availability of peripherals and their number across all MCU series types, refer to the corresponding device data datasheet.

# 52. Revision history

**Table 52-1. Revision history**

| Revision No. | Description | Date |
|---|---|---|
| 1.0 | Initial Release | May. 6, 2023 |
| 1.1 | 1. Update ***HPDF Block Diagram***、***High-Performance Digital Filter (HPDF)*** and ***Threshold monitor*** chapter.<br>2. Update ***Controller area network (CAN)*** chapter overview description.<br>3. Update ***I/O compensation cell code configuration register (SYSCFG_CPSCCCFG)*** register bit field description.<br>4. Update ***Alternate function selected register 0 (GPIOx_AFSEL0, x=A...H, J, K)*** and ***Alternate function selected register 1 (GPIOx_AFSEL1, x=A…H, J, K)*** register bit field description.<br>5. Update ***Figure 20 1. ADC module block diagram.***<br>6. Update the USART module ***Receive / Transmitter FIFO*** chapter description.<br>7. Update GPIO and AFIO module ***Input filtering*** and ***Input filtering register (GPIOx_IFL, x=A...H, J, K) chapter description.***<br>8. Update CPDM module ***Configuration register (CPDM_CFG)*** register bit field description.<br>9. Update CMP module ***Status register (CMP_STAT)*** and ***Interrupt flag clear register (CMP_IFC)*** description.<br>10. Update TRNG module ***Control register (TRNG_CTL)*** register bit field description.<br>11. Update ***Power management unit (PMU)*** chapter description.<br>12. Update ***Global USB control and status register (USBHS_GUSBCS)*** description.<br>13. Update SAI module ***Register definition*** access attribute.<br>14. Update IAP module ***Interrupt flag clear register (IPA_INTC)*** description. | Jul. 21, 2023 |
| 1.2 | 1. Update RTDEC module ***Figure 46-3. Decryption flow with AES-128 in CTR mode***.<br>2. Update ***Trigonometric Math Unit (TMU)*** chapter description.<br>3. Update ***Low power digital temperature sensor (LPDTS*** chapter description.<br>4. Update RSPDIF module ***Figure 33-11. RSPDIF states***.<br>5. Upadate ***Controller area network (CAN)*** chapter | Jan. 4, 2024 |

| | | |
|---|---|---|
| | description.<br>6. Update ***System and memory architecture*** c<br>7. Update RCU module ***Figure 6-2. Clock tree***.<br>8. Update ***Clock configuration register 1 (RCU_CFG1)*** register reset value.<br>9. Update SDIO module ***Figure 36-13. 8-bit data bus width***.<br>10. Update ***Comparator (CMP)*** chapter description. | |
| 1.3 | 1. Update TSCFG4[4:0] bit field description of section ***1.8.21, 1.8.24, 1.8.27***.<br>2. Update ***Power management unit (PMU)*** chapter description.<br>3. Update EXTI module ***Table 8 2. Interrupt vector table.***<br>4. Update ***Trigger selection controller (TRIGSEL)*** chapter description.<br>5. Update GPIO module overview chapter description.<br>6. Update TRNG module ***Figure 12 1. TRNG block diagram.***<br>7. Add ***TMU precision*** description.<br>8. Update DMAMUX module ***Table 18 2. Request multiplexer input mapping***.<br>9. Update ADC module ***Table 20 7. Some examples show the maximum output results for N and M combimations (grayed values indicates truncation，for 12bit ADC).***<br>10. Update ***WWDGT*** register base address.<br>11. Update ***TIMER module Table24-9 、Table24-13 and Table24-17*** description.<br>12. Update ***Inter-integrated circuit interface (I2C)*** module Analog filter delay time<br>13. Update ***TFT-LCD interface (TLI)*** module TLI_LxFLLEN register FLL[13:0] description.<br>14. Update ***Receiver of Sony/Philips Digital Interface (RSPDIF)*** chapter description.<br>15. Update ***Controller area network (CAN)*** chapter description.<br>16. Update ***Real-time decryption (RTDEC)*** chapter description.<br>17. Update ***Hardware semaphore(HWSEM)*** INTEN register reset value, INTC register abbreviations.<br>18. Update ***Universal serial bus High-Speed interface (USBHS)*** chapter 49.7.1、49.7.2 description. | May. 10, 2024 |
| 1.4 | 1. Change the description of USBHS0SW / USBHS1SW to USBHS0SEL/USBHS1SEL in chapter ***6.3.45***.<br>2. Modify the description regarding the output of CMP to the pin section in chapter ***44.3.2***.<br>3. Add new comments to the clock tree in chapter ***6.2.1***. | Nov. 27, 2024 |

| | | |
|---|---|---|
| | 4. Update the description of the SCPF bit for smooth calibration of the RTC in chapter ***23.4.4***.<br>5. When SPICKSS is 10 and 11, modify the description of the sampling point at the rising edge and falling edge in ***Table 46-3***.<br>6. Add a note of caution in ***Table 18-3***.<br>7. When using an external clock source for the SysTick, change the prescaler from dividing by 2 to dividing by 8 in . ***RCU and EXTI*** module.<br>8. Modify the reset value of the CAN_ERR1 register, and swap the functions of bit 15 and bit 14 in chapter ***42.5.6***.<br>9. Change "BRTCF" to "MBTCF" and "BRTCIE" to "MBTCIE" in ***Figure 17-7***.<br>10. Update the AF value corresponding to the pins in ***Table 44-1***.<br>11. Modify the expected coefficients for YUV and YCbCr modes in ***Table 35-5***.<br>12. Add an MDC division factor of 142 division in chapter ***43.3.1***.<br>13. Modify the clock time in ***Table 20-5*** and ***Table 20-6***.<br>14. Modify the read/write attributes of the register bit positions in chapter ***21.4.18***.<br>15. Change the JEDEC 106 ROM address to 0xE00FD000 in ***DBG*** module.<br>16. Modify the read access method of the register and delete the description of PB2 in chater ***10.4***.<br>17. Deleted DVSCFG and its related content, and removed PMU power supply modes 3, 4, and 5 in ***PMU*** module.<br>18. Updated content in ***Figure 18-8***.<br>19. Updated content in chapter ***3.3.5***.<br>20. Updated content in ***SDIO*** module. | |
| 1.5 | 1. Updated the MDMA_CHxSTAT1 register bit 11 BZERR description in ***MDMA chapter 17.4.4*** to include block size or BTLEN+1 considerations.<br>2. Revised the MDMA_CHxCFG register BTLEN note in ***MDMA chapter 17.4.6*** to specify BTLEN+1 must align with DWIDTH and SWIDTH.<br>3. Update the ***PMU module*** to reflect BOR functionality and added a new feature.<br>4. Described baud rate prescaler values for CAN_BT and CAN_FDBT registers in ***CAN chapter 42.5.13*** and ***42.5.30***, using BAUDPSC[9:0] + 1 and DBAUDPSC[9:0] + 1.<br>5. Added a note in ***CAN chapter 42.3.9*** recommending EFDIS bit as 0 to prevent false bus idle detection by enabling edge | Jan. 16, 2025 |

| | | |
|---|---|---|
| | filtering.<br>6. Update the internal counter clock source in the ***CAN*** module and add usage restrictions.<br>7. ***CAN*** module, add instructions for using the filter for flag frames.<br>8. Incorporated a CANCLK configurable feature and guidance on clock source limitations in ***CAN chapter 42.2***, with a referral to GD32H7xx series errata. | |
| 1.6 | 1. Update ***Table 1 5. Details of Boot mode***.<br>2. Update ***Power management unit (PMU)*** chapter description.<br>3. Update PLL all configuration register PLL0VCOSEL/PLL1VCOSEL/PLL2VCOSEL bit descriptions in ***Reset and clock unit (RCU)*** chapter.<br>4. Update Single-enden and differential input channels description in ***Analog-to-digital converter (ADC)*** chapter.<br>5. Update Half-duplex communication mode description in ***Universal synchronous / asynchronous receiver / transmitter (USART)*** chapter.<br>6. Update Programming model in master transmitting mode description in ***Inter-integrated circuit interface (I2C)*** chapter.<br>7. Update I2S disabling sequence description in ***Serial peripheral interface/Inter-IC sound (SPI/I2S)*** chapter.<br>8. Update Control register 0 (SPI_CTL0) MSTART bit description in ***Serial peripheral interface/Inter-IC sound (SPI/I2S)*** chapter.<br>9. Update Data state machine description in ***Secure digital input/output interface (SDIO)*** chapter.<br>10. Update Table 37 1. MDIO pins definition and Pin value register (MDIO_PIN) bits PHYPIN[4:0] description in ***Management data input / output (MDIO)*** chapter.<br>11. Update Function overview description in ***VREF*** chapter.<br>12. Update Baudrate/Counter clock source/Error register 1 (CAN_ERR1) bit 14 and bit 15 description in ***Controller area network (CAN)*** chapter.<br>13. Update Area x configuration register (RTDEC_AREx_CFG) bit 1, Interrupt enable register (RTDEC_INTEN) bit 1 name in ***Real-time decryption (RTDEC)*** chapter.<br>14. Update Lock a semaphore description in ***Hardware semaphore (HWSEM)*** chapter.<br>15. Update Control register (HWSEM_CTLx)(x = 0…31) reset value in ***Hardware semaphore (HWSEM)*** chapter.<br>16. Update Device status register (USBHS_DSTAT) SPST bit | Jul. 16, 2025 |

| | | |
|---|---|---|
| | description in ***Universal serial bus High-Speed interface (USBHS)*** chapter. | |
| 1.7 | 1. Update ADDMSK2[2:0] description and I2C_CTL2 reset value; add note that TIMEOUT register cannot be modified when I2C is enabled; add notes for TIMING register configuration after I2CEN and BYTENUM configuration for SMBus slave transmitter with PEC; add notes for BYTENUM configuration restrictions and START bit modification restrictions in ***Inter-integrated circuit interface (I2C)*** chapter.<br>2. Update CRC configuration description for DZ = 32-bit; update NSS level description when SPI is enabled and MSTART is set; update clock source description for I2S (KERCLK); remove and add I2S serial clock configuration recommendations; add CRC calculation description for SPI slave mode; remove IO23_DRV related description and register bits; remove data bit order description; add note for BIT31 prohibiting AFCTL usage in MFD, MRU and MRB modes; update bit 16 description for software clear condition in ***Serial peripheral interface/Inter-IC sound (SPI/I2S)*** chapter.<br>3. Update TRNG block diagram; change "entropy" to "quality" for random number description in ***True random number generator (TRNG)*** chapter.<br>4. Fix HAU_CTL register bit 5-4 description error (bit flip vs byte flip) in ***Hash Acceleration Unit (HAU)*** chapter.<br>5. Update valid jump edge description for resynchronization (from recessive to dominant) in ***Communication parameters (CAN)*** chapter.<br>6. Update ARE_FMVER register field description; add description that non-encrypted area reads also return 0 during termination event in ***Real-time decryption (RTDEC)*** chapter.<br>7. Update Figure 33.15 RSPDIF overrun error diagram in ***Receiver of Sony/Philips Digital Interface (RSPDIF)*** chapter.<br>8. Swap VCEN and VCRSEL positions in Control register 2 (PMU_CTL2); change "0.9V" to "Vcore" in VCORE power domain description in ***Power management unit (PMU)*** chapter.<br>9. Change MCHxOMCTL to MCHxCOMCTL; set TIMERx_CTL1 bit 7 as reserved in ***TIMER*** chapter.<br>10. Update pin names from MDIO/MDC to MDIO_MDIO/MDIO_MDC in ***Management data*** | Jan. 20, 2026 |

| | | |
|---|---|---|
| | ***input/output (MDIO)*** chapter.<br>11. Update signal names for consistency in ***Digital camera interface (DCI)*** chapter.<br>12. Update EDOUT_OCNT register range description from "-32768 to 32767" to "-32767(0x8001) to 32767(0x7FFF)" in ***Encoder Divided-Output controller (EDOUT)*** chapter.<br>13. Update USART_FCS register bit descriptions (bit 26, 25, 5) for consistency; remove "start bit" condition from receiving sampling noise error description; update frame interval bits from 2.5 to 2 for smart card T=0 mode; add note for USART_STAT REA flag bit reservation in UART3/4/6/7; add note that USART_RT cannot be configured as 0xFFFFFF when oversampling rate is 16; add low-power divider configuration description for IrDA in ***Universal synchronous/asynchronous receiver/transmitter (USART)*** chapter.<br>14. Change ftww to fTWW; update TARG0 and TARG1 access width to 32-bit in Figure 1-8 AXI interconnect matrix block diagram in ***System and memory architecture*** chapter.<br>15. Add bit description for value 0x01 in Sensor T0 data register (LPDTS_SDATA) in ***Low power digital temperature sensor (LPDTS)*** chapter.<br>16. Add note for SAI slave synchronization mode usage in ***Serial Audio Interface (SAI)*** chapter.<br>17. Update PLL VCO clock multiplier description; update FCLK HCLK description in Figure 6-2 Clock tree diagram in ***Reset and clock unit (RCU)*** chapter.<br>18. Add note for Buffer transfer mode; update BTLEN[6:0] description in MDMA_CHxCFG register in ***Master direct memory access controller (MDMA)*** chapter.<br>19. Update GPIO pin default mode description in ***General-purpose and alternate-function I/Os (GPIO and AFIO)*** chapter.<br>20. Update secure JTAG read key description in ***Debug (DBG)*** chapter.<br>21. Update formula 7-2 in ***Clock trim controller (CTC)*** chapter.<br>22. Change DLSTCNT[11:0] to DLSTCNT[6:0] in description in ***Clock phase delay module (CPDM)*** chapter.<br>23. Update formula for serial samples required in threshold monitor fast mode in ***High-Performance Digital Filter (HPDF)*** chapter. | |
| 1.8 | 1. Rename the document to "GD32H73x_75x User Manual" | Mar. 31, 2026 |

| | | |
|---|---|---|
| | and merge the relevant content from the GD32H75E User Manual.<br>2. Remove the description of half-word access to registers in the ***Watchdog Timer (WDGT)*** section.<br>3. Add a note to the Input Filtering subsection of the ***General-purpose and Alternate-function I/Os module (GPIO and AFIO)***.<br>4. In the ***Trigonometric Math Unit (TMU)*** module, remove the ITRTNUM[3:0] field from the Control and Status Register (TMU_CS).<br>5. In the ***True Random Number Generator (TRNG)*** module, add a note for the CONDRST bit field in the Control Register (TRNG_CTL), and add notes for the SEIF and SECS bit fields in the Status Register (TRNG_STAT).<br>6. In the ***System and Memory Architecture*** section, update the description of the TSCFG15 bit field in the TIMERx Configuration Register 2 (SYSCFG_TIMERxCFG2, x=0, 7).<br>7. In the ***Cryptographic Acceleration Unit (CAU)*** section, add a note to the Operating Modes subsection.<br>8. In the ***Controller Area Network (CAN)*** section, update the reset value of the CAN_ERR1 register to 0x000X000X, update the description for exiting Pretended Networking mode, and update the description of the RRFRMS bit in the CAN_CTL2 register.<br>In the ***Analog-to-Digital Converter (ADC)*** section, add instructions for using differential channels and update the software flow for the regular sequence of the ADC. | |

## Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company according to the laws of the People’s Republic of China and other applicable laws. The Company reserves all rights under such laws and no Intellectual Property Rights are transferred (either wholly or partially) or licensed by the Company (either expressly or impliedly) herein. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

To the maximum extent permitted by applicable law, the Company makes no representations or warranties of any kind, express or implied, with regard to the merchantability and the fitness for a particular purpose of the Product, nor does the Company assume any liability arising out of the application or use of any Product. Any information provided in this document is provided only for reference purposes. It is the sole responsibility of the user of this document to determine whether the Product is suitable and fit for its applications and products planned, and properly design, program, and test the functionality and safety of its applications and products planned using the Product. The Product is designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and the Product is not designed or intended for use in (i) safety critical applications such as weapons systems, nuclear facilities, atomic energy controller, combustion controller, aeronautic or aerospace applications, traffic signal instruments, pollution control or hazardous substance management; (ii) life-support systems, other medical equipment or systems (including life support equipment and surgical implants); (iii) automotive applications or environments, including but not limited to applications for active and passive safety of automobiles (regardless of front market or aftermarket), for example, EPS, braking, ADAS (camera/fusion), EMS, TCU, BMS, BSG, TPMS, Airbag, Suspension, DMS, ICMS, Domain, ESC, DCDC, e-clutch, advanced-lighting, etc.. Automobile herein means a vehicle propelled by a self-contained motor, engine or the like, such as, without limitation, cars, trucks, motorcycles, electric cars, and other transportation devices; and/or (iv) other uses where the failure of the device or the Product can reasonably be expected to result in personal injury, death, or severe property or environmental damage (collectively "Unintended Uses"). Customers shall take any and all actions to ensure the Product meets the applicable laws and regulations. The Company is not liable for, in whole or in part, and customers shall hereby release the Company as well as its suppliers and/or distributors from, any claim, damage, or other liability arising from or related to all Unintended Uses of the Product. Customers shall indemnify and hold the Company, and its officers, employees, subsidiaries, affiliates as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Product.

Information in this document is provided solely in connection with the Product. The Company reserves the right to make changes, corrections, modifications or improvements to this document and the Product described herein at any time without notice. The Company shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 GigaDevice Semiconductor Inc. – All rights reserved