GigaDevice Semiconductor Inc.

GD32F527xx
Arm® Cortex®-M33 32-bit MCU

User Manual

Revision 1.5

(Sep. 2026)

Table of Contents

Table of Contents **2**

List of Figures **25**

List of Tables **35**

1. System and memory architecture **40**

1.1. Arm® Cortex®-M33 processor **40**

1.2. System architecture **41**

1.3. Memory map **43**

1.3.1. On-chip SRAM memory 48

1.3.2. On-chip Flash memory 49

1.4. Boot configuration **49**

1.5. System configuration controller **51**

1.6. System configuration registers **52**

1.6.1. Configuration register 0 (SYSCFG_CFG0) 52

1.6.2. Configuration register 1 (SYSCFG_CFG1) 53

1.6.3. EXTI sources selection register 0 (SYSCFG_EXTISS0) 54

1.6.4. EXTI sources selection register 1 (SYSCFG_EXTISS1) 55

1.6.5. EXTI sources selection register 2 (SYSCFG_EXTISS2) 56

1.6.6. EXTI sources selection register 3 (SYSCFG_EXTISS3) 58

1.6.7. Compensation cell control register (SYSCFG_CPSCTL) 59

1.6.8. System status register (SYSCFG_STAT) 60

1.6.9. SRAM0 ECC status register (SYSCFG_SRAM0ECC) 62

1.6.10. SRAM1 ECC status register (SYSCFG_SRAM1ECC) 62

1.6.11. SRAM2 ECC status register (SYSCFG_SRAM2ECC) 63

1.6.12. ADDSRAM ECC status register (SYSCFG_ADDSRAMECC) 64

1.6.13. TCMSRAM ECC register (SYSCFG_TCMSRAMECC) 65

1.6.14. BKPSRAM ECC register (SYSCFG_BKPSRAMECC) 66

1.6.15. FLASH ECC address register (SYSCFG_FLASHECC_ADDR) 66

1.6.16. FLASH ECC register (SYSCFG_FLASHECC) 67

1.6.17. User configuration register (USER_CFG) 67

1.7. Device electronic signature **68**

1.7.1. Memory density information 68

1.7.2. Unique device ID (96 bits) 68

2. System Security **70**

2.1. Overview **70**

2.2. Characteristics **70**

2.3. Memory security 71
2.3.1. System Flash protection 71
2.3.2. User Flash protection 71
2.3.3. SRAM protection 73
2.3.4. Trusted code protection 73
2.3.5. Password protection 74
2.3.6. External memory protection 75
2.4. Boot protection 75
2.4.1. Unique boot entry 76
2.4.2. Secure boot 76
2.5. Debugging security 77
2.5.1. Limit debugging 77
2.5.2. Disable debugging function 78
2.6. Encryption and random numbers 78
2.7. System monitoring 79
2.7.1. Tamper protection (with RTC) 79
2.7.2. Power supply supervision 79
2.7.3. Clock security system 79
2.7.4. Temperature sensor 80
2.7.5. Critical code execution time monitoring 80
2.8. Device UID 80
3. Flash memory controller (FMC) 81
3.1. Overview 81
3.2. Characteristics 81
3.3. Function overview 81
3.3.1. Flash memory architecture 81
3.3.2. Error Checking and Correcting (ECC) 85
3.3.3. Read operations 85
3.3.4. Unlock the FMC_CTL/FMC_OBCTLx register 86
3.3.5. Page erase 86
3.3.6. Sector erase 87
3.3.7. Mass erase 88
3.3.8. Main flash programming 90
3.3.9. OTP block programming 91
3.3.10. Option bytes modify 93
3.3.11. Option bytes description 93
3.3.12. Sector erase/program protection 95
3.3.13. CBUS read protection 96
3.3.14. Security protection 96
3.3.15. EFUSE macro description 97
3.3.16. EFUSE read operation 98

3.3.17. EFUSE program operation 98
3.4. Register definition 99
3.4.1. Unlock key register (FMC_KEY) 99
3.4.2. Option byte unlock key register (FMC_OBKEY) 99
3.4.3. Status register (FMC_STAT) 99
3.4.4. Control register (FMC_CTL) 101
3.4.5. Option byte control register 0 (FMC_OBCTL0) 103
3.4.6. Option byte control register 1 (FMC_OBCTL1) 104
3.4.7. Page erase configuration register (FMC_PECFG) 105
3.4.8. Unlock page erase key register (FMC_PEKEY) 106
3.4.9. OTP1 configuration register (FMC_OTP1CFG) 106
3.4.10. Load code ECC error address0 (FMC_LDECCADDR0) 107
3.4.11. Load code ECC error address1 (FMC_LDECCADDR1) 107
3.4.12. Load code ECC error address2 (FMC_LDECCADDR2) 107
3.4.13. Option bytes status register (FMC_OBSTAT) 108
3.4.14. Product ID register (FMC_PID) 108
3.4.15. EFUSE control and status register (EFUSE_CS) 109
3.4.16. EFUSE address register (EFUSE_ADDR) 110
3.4.17. EFUSE control register (EFUSE_CTL) 111
3.4.18. EFUSE user data register (EFUSE_USER_DATA) 112
4. Power management unit (PMU) 113
4.1. Overview 113
4.2. Characteristics 113
4.3. Function overview 113
4.3.1. Battery backup domain 114
4.3.2. Backup SRAM 115
4.3.3. V_{DD} / V_{DDA} power domain 115
4.3.4. V_{CORE} power domain 118
4.3.5. Power saving modes 119
4.4. Register definition 122
4.4.1. Control register (PMU_CTL) 122
4.4.2. Control and status register (PMU_CS) 124
5. Reset and clock unit (RCU) 126
5.1. Reset control unit (RCTL) 126
5.1.1. Overview 126
5.1.2. Function overview 126
5.2. Clock control unit (CCTL) 127
5.2.1. Overview 127
5.2.2. Characteristics 129
5.2.3. Function overview 130

5.3. Register definition 135
5.3.1. Control register (RCU_CTL) 135
5.3.2. PLL register (RCU_PLL) 137
5.3.3. Clock configuration register 0 (RCU_CFG0) 139
5.3.4. Clock interrupt register (RCU_INT) 141
5.3.5. AHB1 reset register (RCU_AHB1RST) 144
5.3.6. AHB2 reset register (RCU_AHB2RST) 146
5.3.7. AHB3 reset register (RCU_AHB3RST) 147
5.3.8. APB1 reset register (RCU_APB1RST) 148
5.3.9. APB2 reset register (RCU_APB2RST) 151
5.3.10. AHB1 enable register (RCU_AHB1EN) 153
5.3.11. AHB2 enable register (RCU_AHB2EN) 156
5.3.12. AHB3 enable register (RCU_AHB3EN) 157
5.3.13. APB1 enable register (RCU_APB1EN) 158
5.3.14. APB2 enable register (RCU_APB2EN) 161
5.3.15. AHB1 sleep mode enable register (RCU_AHB1SPEN) 163
5.3.16. AHB2 sleep mode enable register (RCU_AHB2SPEN) 166
5.3.17. AHB3 sleep mode enable register (RCU_AHB3SPEN) 167
5.3.18. APB1 sleep mode enable register (RCU_APB1SPEN) 168
5.3.19. APB2 sleep mode enable register (RCU_APB2SPEN) 171
5.3.20. Backup domain control register (RCU_BDCTL) 174
5.3.21. Reset source/clock register (RCU_RSTSCK) 175
5.3.22. PLL clock spread spectrum control register (RCU_PLLSSCTL) 177
5.3.23. PLLI2S register (RCU_PLLI2S) 177
5.3.24. PLLSAI register (RCU_PLLSAI) 179
5.3.25. Clock configuration register 1 (RCU_CFG1) 181
5.3.26. Clock configuration register 2 (RCU_CFG2) 182
5.3.27. Additional clock control register (RCU_ADDCTL) 183
5.3.28. Additional clock interrupt register (RCU_ADDINT) 184
5.3.29. APB1 additional reset register (RCU_ADDAPB1RST) 185
5.3.30. APB1 additional enable register (RCU_ADDAPB1EN) 185
5.3.31. APB1 additional sleep mode enable register (RCU_ADDAPB1SPEN) 186
5.3.32. Voltage key register (RCU_VKEY) 187
5.3.33. Deep-sleep mode voltage register (RCU_DSV) 187
6. Clock trim controller (CTC) 189
6.1. Overview 189
6.2. Characteristics 189
6.3. Function overview 189
6.3.1. Reference sync pulse generator 190
6.3.2. CTC trim counter 190
6.3.3. Frequency evaluation and automatic trim process 191
6.3.4. Software program guide 192

6.4. Register definition 193
6.4.1. Control register 0 (CTC_CTL0) 193
6.4.2. Control register 1 (CTC_CTL1) 194
6.4.3. Status register (CTC_STAT) 195
6.4.4. Interrupt clear register (CTC_INTC) 197
7. Interrupt / event controller (EXTI) 199
7.1. Overview 199
7.2. Characteristics 199
7.3. Function overview 199
7.4. External interrupt and event block diagram 203
7.5. External Interrupt and Event function overview 204
7.6. Register definition 206
7.6.1. Interrupt enable register (EXTI_INTEN) 206
7.6.2. Event enable register (EXTI_EVEN) 206
7.6.3. Rising edge trigger enable register (EXTI_RTEN) 207
7.6.4. Falling edge trigger enable register (EXTI_FTEN) 207
7.6.5. Software interrupt event register (EXTI_SWIEV) 207
7.6.6. Pending register (EXTI_PD) 208
8. General-purpose and alternate-function I/Os (GPIO and AFIO) 209
8.1. Overview 209
8.2. Characteristics 209
8.3. Function overview 209
8.3.1. GPIO pin configuration 211
8.3.2. External interrupt / event lines 211
8.3.3. Alternate functions (AF) 211
8.3.4. Additional functions 211
8.3.5. Input configuration 212
8.3.6. Output configuration 212
8.3.7. Analog configuration 213
8.3.8. Alternate function (AF) configuration 213
8.3.9. GPIO locking function 214
8.3.10. GPIO single cycle toggle function 214
8.4. Register definition 215
8.4.1. Port control register (GPIOx_CTL, x = A…I) 215
8.4.2. Port output mode register (GPIOx_OMODE, x = A…I) 217
8.4.3. Port output speed register (GPIOx_OSPD, x = A…I) 218
8.4.4. Port pull-up / pull-down register (GPIOx_PUD, x = A…I) 220
8.4.5. Port input status register (GPIOx_ISTAT, x = A…I) 222
8.4.6. Port output control register (GPIOx_OCTL, x = A…I) 222
8.4.7. Port bit operate register (GPIOx_BOP, x = A…I) 223

8.4.8. Port configuration lock register (GPIOx_LOCK, x = A…I) 223
8.4.9. Alternate function selected register 0 (GPIOx_AFSEL0, x = A…I) 224
8.4.10. Alternate function selected register 1 (GPIOx_AFSEL1, x = A…I) 225
8.4.11. Bit clear register (GPIOx_BC, x = A…I) 226
8.4.12. Port bit toggle register (GPIOx_TG, x = A…I) 226
9. Cyclic redundancy checks management unit (CRC) 228
9.1. Overview 228
9.2. Characteristics 228
9.3. Function overview 229
9.4. Register definition 230
9.4.1. Data register (CRC_DATA) 230
9.4.2. Free data register (CRC_FDATA) 230
9.4.3. Control register (CRC_CTL) 231
10. True random number generator (TRNG) 232
10.1. Overview 232
10.2. Characteristics 232
10.3. Function overview 232
10.3.1. Operation flow 233
10.3.2. Error flags 233
10.4. Register definition 234
10.4.1. Control register (TRNG_CTL) 234
10.4.2. Status register (TRNG_STAT) 234
10.4.3. Data register (TRNG_DATA) 235
11. Public Key Cryptographic Acceleration Unit (PKCAU) 237
11.1. Overview 237
11.2. Characteristics 237
11.3. Function overview 237
11.3.1. Operands 238
11.3.2. RSA algorithm 238
11.3.3. ECC algorithm 240
11.3.4. Integer arithmetic operations 241
11.3.5. Elliptic curve operations in Fp domain 251
11.3.6. PKCAU operation process 257
11.3.7. Processing times 258
11.3.8. Status, errors and interrupts 259
11.4. Register definition 261
11.4.1. Control register (PKCAU_CTL) 261
11.4.2. Status register (PKCAU_STAT) 262
11.4.3. Status clear register (PKCAU_STATC) 263

12. Hash Acceleration Unit (HAU) 264
12.1. Overview 264
12.2. Characteristics 264
12.3. HAU data type 264
12.4. HAU core 266
12.4.1. Automatic data padding 266
12.4.2. Digest computing 267
12.4.3. Hash mode 268
12.4.4. HMAC mode 268
12.5. HAU suspended mode 268
12.5.1. Transfer data by CPU 269
12.5.2. Transfer data by DMA 269
12.6. HAU interrupt 270
12.6.1. Input FIFO interrupt 270
12.6.2. Calculation completion interrupt 270
12.7. Register definition 271
12.7.1. HAU control register (HAU_CTL) 271
12.7.2. HAU data input register (HAU_DI) 272
12.7.3. HAU configuration register (HAU_CFG) 273
12.7.4. HAU data output register (HAU_DO0..7) 274
12.7.5. HAU interrupt enable register (HAU_INTEN) 276
12.7.6. HAU status and flag register (HAU_STAT) 276
12.7.7. Context switch register x (HAU_CTXSx) (x = 0...53) 277
13. Cryptographic Acceleration Unit (CAU) 278
13.1. Overview 278
13.2. Characteristics 278
13.3. CAU data type and initialization vectors 279
13.3.1. Data type 279
13.3.2. Initialization vectors 280
13.4. Cryptographic acceleration processor 280
13.4.1. DES / TDES cryptographic acceleration processor 281
13.4.2. AES cryptographic acceleration processor 285
13.5. Operating modes 292
13.6. CAU DMA interface 293
13.7. CAU interrupts 294
13.8. CAU suspended mode 294
13.9. Register definition 296
13.9.1. Control register (CAU_CTL) 296

13.9.2. Status register 0 (CAU_STAT0) 298
13.9.3. Data input register (CAU_DI) 298
13.9.4. Data output register (CAU_DO) 299
13.9.5. DMA enable register (CAU_DMAEN) 299
13.9.6. Interrupt enable register (CAU_INTEN) 300
13.9.7. Status register 1 (CAU_STAT1) 300
13.9.8. Interrupt flag register (CAU_INTF) 301
13.9.9. Key registers (CAU_KEY0..3 (H / L)) 302
13.9.10. Initial vector registers (CAU_IV0..1 (H / L)) 304
13.9.11. GCM or CCM mode context switch register x (CAU_GCMCCMCTXSx) (x = 0..7) 306
13.9.12. GCM mode context switch register x (CAU_GCMCTXSx) (x = 0..7) 306

14. Direct memory access controller (DMA) 307

14.1. Overview 307

14.2. Characteristics 307

14.3. Block diagram 308

14.4. Function overview 308

14.4.1. Peripheral handshake 310
14.4.2. Data process 311
14.4.3. Address generation 317
14.4.4. Circular mode 317
14.4.5. Switch-buffer mode 318
14.4.6. Transfer flow controller 318
14.4.7. Transfer operation 319
14.4.8. Transfer finish 320
14.4.9. Channel configuration 321

14.5. Interrupts 322

14.5.1. Flag 323
14.5.2. Exception 324
14.5.3. Error 324

14.6. Register definition 327

14.6.1. Interrupt flag register 0 (DMA_INTF0) 327
14.6.2. Interrupt flag register 1 (DMA_INTF1) 328
14.6.3. Interrupt flag clear register 0 (DMA_INTC0) 329
14.6.4. Interrupt flag clear register 1 (DMA_INTC1) 329
14.6.5. Channel x control register (DMA_CHxCTL) 330
14.6.6. Channel x counter register (DMA_CHxCNT) 334
14.6.7. Channel x peripheral base address register (DMA_CHxPADDR) 334
14.6.8. Channel x memory 0 base address register (DMA_CHxM0ADDR) 335
14.6.9. Channel x memory 1 base address register (DMA_CHxM1ADDR) 336
14.6.10. Channel x FIFO control register (DMA_CHxFCTL) 336

15. Image processing accelerator (IPA) 338

15.1. Overview 338
15.2. Characteristics 338
15.3. Block diagram 339
15.4. Function overview 339
15.4.1. Conversion operation 340
15.4.2. Foreground and background LUT 341
15.4.3. Foreground and background pixel channel extension (PCE) 342
15.4.4. Blending 344
15.4.5. Destination pixel channel compression (PCC) 345
15.4.6. Inter-timer 346
15.4.7. Line mark 347
15.4.8. Transfer flow 347
15.4.9. Configuration 348
15.5. Interrupts 351
15.6. Register definition 355
15.6.1. Control register (IPA_CTL) 355
15.6.2. Interrupt flag register (IPA_INTF) 356
15.6.3. Interrupt flag clear register (IPA_INTC) 357
15.6.4. Foreground memory base address register (IPA_FMADDR) 358
15.6.5. Foreground line offset register (IPA_FLOFF) 359
15.6.6. Background memory base address register (IPA_BMADDR) 359
15.6.7. Background line offset register (IPA_BLOFF) 360
15.6.8. Foreground pixel control register (IPA_FPCTL) 360
15.6.9. Foreground pixel value register (IPA_FPV) 362
15.6.10. Background pixel control register (IPA_BPCTL) 363
15.6.11. Background pixel value register (IPA_BPV) 364
15.6.12. Foreground LUT memory base address register (IPA_FLMADDR) 365
15.6.13. Background LUT memory base address register (IPA_BLMADDR) 366
15.6.14. Destination pixel control register (IPA_DPCTL) 366
15.6.15. Destination pixel value register (IPA_DPV) 367
15.6.16. Destination memory base address register (IPA_DMADDR) 370
15.6.17. Destination line offset register (IPA_DLOFF) 370
15.6.18. Image size register (IPA_IMS) 371
15.6.19. Line mark register (IPA_LM) 372
15.6.20. Inter-timer control register (IPA_ITCTL) 372
16. Debug (DBG) 374
16.1. Overview 374
16.2. JTAG/SW function overview 374
16.3. Switch JTAG or SW interface 374
16.3.1. Pin assignment 374
16.3.2. JTAG daisy chained structure 375

16.3.3. Debug reset 375
16.3.4. JEDEC-106 ID code 375
16.4. Debug hold function overview 375
16.4.1. Debug support for power saving mode 375
16.4.2. Debug support for TIMER, I2C, RTC, WWDGT, FWDGT and CAN 376
16.5. Register definition 377
16.5.1. ID code register (DBG_ID) 377
16.5.2. Control register 0 (DBG_CTL0) 377
16.5.3. Control register 1 (DBG_CTL1) 378
16.5.4. Control register 2 (DBG_CTL2) 380
16.5.5. Control register 3 (DBG_CTL3) 382
17. Programmable current reference (IREF) 383
17.1. Overview 383
17.2. Characteristics 383
17.3. Function overview 383
17.3.1. Signal description 383
17.3.2. User trimming 383
17.4. Register definition 384
17.4.1. Control register (IREF_CTL) 384
18. Analog-to-digital converter (ADC) 385
18.1. Overview 385
18.2. Characteristics 385
18.3. Pins and internal signals 386
18.4. Function overview 387
18.4.1. Foreground calibration function 387
18.4.2. ADC clock 388
18.4.3. ADCON enable 388
18.4.4. Routine sequence 388
18.4.5. Operation modes 388
18.4.6. Conversion result threshold monitor function 391
18.4.7. Data storage mode 391
18.4.8. Sample time configuration 392
18.4.9. External trigger configuration 393
18.4.10. DMA request 393
18.4.11. Overflow detection 394
18.4.12. ADC internal channels 394
18.4.13. Battery voltage monitoring 395
18.4.14. Programmable resolution (DRES) 395
18.4.15. On-chip hardware oversampling 395

18.5. ADC sync mode 397
18.5.1. Free mode 398
18.5.2. Routine parallel mode 398
18.5.3. Routine follow-up mode 399
18.5.4. Use DMA in ADC sync mode 400
18.6. ADC interrupts 401
18.7. Register definition 402
18.7.1. Status register (ADC_STAT) 402
18.7.2. Control register 0 (ADC_CTL0) 403
18.7.3. Control register 1 (ADC_CTL1) 405
18.7.4. Sample time register 0 (ADC_SAMPT0) 406
18.7.5. Sample time register 1 (ADC_SAMPT1) 407
18.7.6. Watchdog high threshold register (ADC_WDHT) 408
18.7.7. Watchdog low threshold register (ADC_WDLT) 409
18.7.8. Routine sequence register 0 (ADC_RSQ0) 409
18.7.9. Routine sequence register 1 (ADC_RSQ1) 410
18.7.10. Routine sequence register 2 (ADC_RSQ2) 410
18.7.11. Routine data register (ADC_RDATA) 411
18.7.12. Oversample control register (ADC_OVSAMPCTL) 411
18.7.13. Summary status register (ADC_SSTAT) 412
18.7.14. Sync control register (ADC_SYNCCTL) 413
18.7.15. Sync routine data register (ADC_SYNCDATA) 415
19. Digital-to-analog converter (DAC) 416
19.1. Overview 416
19.2. Characteristics 416
19.3. Function overview 418
19.3.1. DAC enable 418
19.3.2. DAC output buffer 418
19.3.3. DAC data configuration 418
19.3.4. DAC trigger 418
19.3.5. DAC conversion 419
19.3.6. DAC noise wave 419
19.3.7. DAC output voltage 420
19.3.8. DMA request 420
19.3.9. DAC concurrent conversion 420
19.4. Register definition 422
19.4.1. DACx control register 0 (DAC_CTL0) 422
19.4.2. DACx software trigger register (DAC_SWT) 425
19.4.3. DACx_OUT0 12-bit right-aligned data holding register (DAC_OUT0_R12DH) 425
19.4.4. DACx_OUT0 12-bit left-aligned data holding register (DAC_OUT0_L12DH) 426
19.4.5. DACx_OUT0 8-bit right-aligned data holding register (DAC_OUT0_R8DH) 426
19.4.6. DACx_OUT1 12-bit right-aligned data holding register (DAC_OUT1_R12DH) 426

19.4.7. DACx_OUT1 12-bit left-aligned data holding register (DAC_OUT1_L12DH) 427
19.4.8. DACx_OUT1 8-bit right-aligned data holding register (DAC_OUT1_R8DH) 427
19.4.9. DACx concurrent mode 12-bit right-aligned data holding register (DACC_R12DH) 428
19.4.10. DACx concurrent mode 12-bit left-aligned data holding register (DACC_L12DH) 428
19.4.11. DACx concurrent mode 8-bit right-aligned data holding register (DACC_R8DH) 429
19.4.12. DACx_OUT0 data output register (DAC_OUT0_DO) 429
19.4.13. DACx_OUT1 data output register (DAC_OUT1_DO) 430
19.4.14. DACx status register 0 (DAC_STAT0) 430

20. Watchdog timer (WDGT) 432

20.1. Free watchdog timer (FWDGT) 432
20.1.1. Overview 432
20.1.2. Characteristics 432
20.1.3. Function overview 432
20.1.4. Register definition 435

20.2. Window watchdog timer (WWDGT) 438
20.2.1. Overview 438
20.2.2. Characteristics 438
20.2.3. Function overview 438
20.2.4. Register definition 441

21. Real time clock (RTC) 443

21.1. Overview 443

21.2. Characteristics 443

21.3. Function overview 444
21.3.1. Block diagram 444
21.3.2. Clock source and prescalers 444
21.3.3. Shadow registers introduction 445
21.3.4. Configurable and field maskable alarm 445
21.3.5. Configurable periodic auto-wakeup counter 446
21.3.6. RTC initialization and configuration 446
21.3.7. Calendar reading 447
21.3.8. Resetting the RTC 448
21.3.9. RTC shift function 449
21.3.10. RTC reference clock detection 450
21.3.11. RTC coarse digital calibration 450
21.3.12. RTC smooth digital calibration 451
21.3.13. Time-stamp function 453
21.3.14. Tamper detection 453
21.3.15. Calibration clock output 454
21.3.16. Alarm output 455
21.3.17. RTC power saving mode management 455
21.3.18. RTC interrupts 455

21.4. Register definition 456
21.4.1. Time register (RTC_TIME) 456
21.4.2. Date register (RTC_DATE) 456
21.4.3. Control register (RTC_CTL) 457
21.4.4. Status register (RTC_STAT) 460
21.4.5. Prescaler register (RTC_PSC) 462
21.4.6. Wakeup timer register (RTC_WUT) 462
21.4.7. Coarse calibration register (RTC_COSC) 463
21.4.8. Alarm 0 time and date register (RTC_ALRM0TD) 463
21.4.9. Alarm 1 time and date register (RTC_ALRM1TD) 464
21.4.10. Write protection key register (RTC_WPK) 466
21.4.11. Sub second register (RTC_SS) 466
21.4.12. Shift function control register (RTC_SHIFTCTL) 466
21.4.13. Time of time stamp register (RTC_TTS) 467
21.4.14. Date of time stamp register (RTC_DTS) 468
21.4.15. Sub second of time stamp register (RTC_SSTS) 468
21.4.16. High resolution frequency compensation register (RTC_HRFC) 469
21.4.17. Tamper register (RTC_TAMP) 470
21.4.18. Alarm 0 sub second register (RTC_ALRM0SS) 472
21.4.19. Alarm 1 sub second register (RTC_ALRM1SS) 473
21.4.20. Backup registers (RTC_BKPx) (x=0..19) 474

22. Timer (TIMERx) 475

22.1. Advanced timer (TIMERx, x=0, 7) 476
22.1.1. Overview 476
22.1.2. Characteristics 476
22.1.3. Function overview 477
22.1.4. Register definition 509

22.2. General level0 timer (TIMERx, x=1, 2, 3, 4) 541
22.2.1. Overview 541
22.2.2. Characteristics 541
22.2.3. Function overview 541
22.2.4. Register definition 556

22.3. General level1 timer (TIMERx, x=8, 11) 582
22.3.1. Overview 582
22.3.2. Characteristics 582
22.3.3. Function overview 583
22.3.4. Register definition 595

22.4. General level2 timer (TIMERx, x=9, 10, 12, 13) 608
22.4.1. Overview 608
22.4.2. Characteristics 608
22.4.3. Function overview 608
22.4.4. Register definition 616

22.5. Basic timer (TIMERx, x=5, 6) 627
22.5.1. Overview 627
22.5.2. Characteristics 627
22.5.3. Function overview 627
22.5.4. Register definition 631
23. Universal synchronous/asynchronous receiver /transmitter (USART) 636
23.1. Overview 636
23.2. Characteristics 636
23.3. Function overview 637
23.3.1. USART frame format 638
23.3.2. Baud rate generation 639
23.3.3. USART transmitter 639
23.3.4. USART receiver 641
23.3.5. Use DMA for data buffer access 642
23.3.6. Hardware flow control 644
23.3.7. Multi-processor communication 645
23.3.8. LIN mode 646
23.3.9. Synchronous mode 647
23.3.10. IrDA SIR ENDEC mode 648
23.3.11. Half-duplex communication mode 649
23.3.12. Smartcard (ISO7816-3) mode 649
23.3.13. USART interrupts 651
23.4. Register definition 653
23.4.1. Status register 0 (USART_STAT0) 653
23.4.2. Data register (USART_DATA) 655
23.4.3. Baud rate register (USART_BAUD) 655
23.4.4. Control register 0 (USART_CTL0) 656
23.4.5. Control register 1 (USART_CTL1) 658
23.4.6. Control register 2 (USART_CTL2) 659
23.4.7. Guard time and prescaler register (USART_GP) 661
23.4.8. Control register 3 (USART_CTL3) 662
23.4.9. Receiver timeout register (USART_RT) 663
23.4.10. Status register 1 (USART_STAT1) 664
23.4.11. Coherence control register (USART_CHC) 665
24. Inter-integrated circuit interface (I2C) 667
24.1. Inter-integrated circuit interface (I2Cx, x=0, 1, 2) 667
24.1.1. Overview 667
24.1.2. Characteristics 667
24.1.3. Function overview 667
24.1.4. Register definition 684
24.2. Inter-integrated circuit interface (I2Cx, x=3, 4, 5) 696

24.2.1. Overview 696
24.2.2. Characteristics 696
24.2.3. Function overview 696
24.2.4. Register definition 722
25. Serial peripheral interface/Inter-IC sound (SPI/I2S) 736
25.1. Overview 736
25.2. Characteristics 736
25.2.1. SPI characteristics 736
25.2.2. I2S characteristics 736
25.3. SPI block diagram 737
25.4. SPI signal description 737
25.4.1. Normal configuration (Not Quad-SPI Mode) 737
25.4.2. Quad-SPI configuration 738
25.5. SPI function overview 738
25.5.1. SPI clock timing and data format 738
25.5.2. NSS function 739
25.5.3. SPI operation modes 740
25.5.4. DMA function 748
25.5.5. CRC function 748
25.6. SPI interrupts 749
25.6.1. Status flags 749
25.6.2. Error conditions 749
25.7. I2S block diagram 750
25.8. I2S signal description 751
25.9. I2S function overview 751
25.9.1. I2S audio standards 751
25.9.2. I2S clock 759
25.9.3. Operation 760
25.9.4. DMA function 763
25.10. I2S interrupts 764
25.10.1. Status flags 764
25.10.2. Error conditions 764
25.11. Register definition 766
25.11.1. Control register 0 (SPI_CTL0) 766
25.11.2. Control register 1 (SPI_CTL1) 768
25.11.3. Status register (SPI_STAT) 769
25.11.4. Data register (SPI_DATA) 770
25.11.5. CRC polynomial register (SPI_CRCPOLY) 771
25.11.6. RX CRC register (SPI_RCRC) 771
25.11.7. TX CRC register (SPI_TCRC) 772

25.11.8. I2S control register (SPI_I2SCTL) 773
25.11.9. I2S clock prescaler register (SPI_I2SPSC) 774
25.11.10. Quad-SPI mode control register (SPI_QCTL) of SPI5 775
26. Serial Audio Interface (SAI) 776
26.1. Overview 776
26.2. Characteristics 776
26.3. Function overview 777
26.3.1. Block diagram 777
26.3.2. Clock divider 778
26.3.3. Operating mode 779
26.3.4. SAI synchronization mode 780
26.3.5. Frame configuration 781
26.3.6. Slot configuration 782
26.3.7. Data configuration 785
26.3.8. Internal FIFO 785
26.3.9. AC'97 link controller 786
26.3.10. SPDIF Output 787
26.3.11. Stereo/Mono 789
26.3.12. Mute 789
26.3.13. Compander 790
26.3.14. Output drive 793
26.3.15. IO management 793
26.3.16. DMA interface 793
26.3.17. Enable/Disable 794
26.3.18. Error flags 794
26.3.19. Interrupts 797
26.4. Register definition 798
26.4.1. Synchronize configuration register (SAI_SYNCFG) 798
26.4.2. Block x configuration register0 (SAI_BxCFG0) (x = 0,1) 798
26.4.3. Block x configuration register1 (SAI_BxCFG1) (x = 0,1) 801
26.4.4. Block x frame configuration register (SAI_BxFCFG) (x = 0,1) 803
26.4.5. Block x slot configuration register (SAI_BxSCFG) (x = 0,1) 804
26.4.6. Block x interrupt enable register (SAI_BxINTEN) (x = 0,1) 805
26.4.7. Block x status register (SAI_BxSTAT) (x = 0,1) 807
26.4.8. Block x interrupt flag clear register (SAI_BxINTC) (x = 0,1) 809
26.4.9. Block x data register (SAI_BxDATA) (x = 0,1) 810
27. Digital camera interface (DCI) 811
27.1. Overview 811
27.2. Characteristics 811
27.3. Block diagram 811

27.4. Signal description 812
27.5. Function overview 812
27.5.1. DCI hardware synchronization mode 812
27.5.2. Embedded synchronization mode 813
27.5.3. Capture data using snapshot or continuous capture modes 813
27.5.4. Window function 814
27.5.5. Pixel formats, data padding and DMA 814
27.6. Interrupts 815
27.7. Register definition 816
27.7.1. Control register (DCI_CTL) 816
27.7.2. Status register0 (DCI_STAT0) 817
27.7.3. Status register1 (DCI_STAT1) 818
27.7.4. Interrupt enable register (DCI_INTEN) 818
27.7.5. Interrupt flag register (DCI_INTF) 819
27.7.6. Interrupt flag clear register (DCI_INTC) 820
27.7.7. Synchronization codes register (DCI_SC) 820
27.7.8. Synchronization codes unmask register (DCI_SCUMSK) 821
27.7.9. Cropping window start position register (DCI_CWSPOS) 821
27.7.10. Cropping window size register (DCI_CWSZ) 822
27.7.11. DATA register (DCI_DATA) 822
28. TFT-LCD interface (TLI) 824
28.1. Overview 824
28.2. Characteristics 824
28.3. Block diagram 824
28.4. Signal description 825
28.5. Function overview 825
28.5.1. LCD display timing 825
28.5.2. Pixel DMA function 826
28.5.3. Pixel formats 827
28.5.4. Layer window and blending function 827
28.5.5. Layer configuration reload 828
28.5.6. Dithering function 829
28.6. Interrupts 829
28.7. Register definition 830
28.7.1. Synchronous pulse size register (TLI_SPSZ) 830
28.7.2. Back-porch size register (TLI_BPSZ) 830
28.7.3. Active size register (TLI_ASZ) 831
28.7.4. Total size register (TLI_TSZ) 831
28.7.5. Control register (TLI_CTL) 832
28.7.6. Reload layer register (TLI_RL) 833

28.7.7. Background color register (TLI_BGC) 834
28.7.8. Interrupt enable register (TLI_INTEN) 834
28.7.9. Interrupt flag register (TLI_INTF) 835
28.7.10. Interrupt flag clear register (TLI_INTC) 835
28.7.11. Line mark register (TLI_LM) 836
28.7.12. Current pixel position register (TLI_CPPOS) 836
28.7.13. Status register (TLI_STAT) 837
28.7.14. Layer x control register (TLI_LxCTL) (x = 0, 1) 837
28.7.15. Layer x horizontal position parameters register (TLI_LxHPOS) (x = 0, 1) 838
28.7.16. Layer x vertical position parameters register (TLI_LxVPOS) (x = 0, 1) 838
28.7.17. Layer x color key register (TLI_LxCKEY) (x = 0, 1) 839
28.7.18. Layer x packeted pixel format register (TLI_LxPPF) (x = 0, 1) 839
28.7.19. Layer x specified alpha register (TLI_LxSA) (x = 0, 1) 840
28.7.20. Layer x default color register (TLI_LxDC) (x = 0, 1) 840
28.7.21. Layer x blending register (TLI_LxBLEND) (x = 0, 1) 841
28.7.22. Layer x frame base address register (TLI_LxFBADDR) (x = 0, 1) 842
28.7.23. Layer x frame line length register (TLI_LxFLLEN) (x = 0, 1) 842
28.7.24. Layer x frame total line number register (TLI_LxFTLN) (x = 0, 1) 843
28.7.25. Layer x look up table register (TLI_LxLUT) (x = 0, 1) 843

29. Secure digital input/output interface (SDIO) 844

29.1. Introduction 844

29.2. Main features 844

29.3. SDIO bus topology 844

29.4. SDIO functional description 847

29.4.1. SDIO adapter 847
29.4.2. APB2 interface 851

29.5. Card functional description 853

29.5.1. Card registers 853
29.5.2. Commands 854
29.5.3. Responses 865
29.5.4. Data packets format 869
29.5.5. Two status fields of the card 870

29.6. Programming sequence 877

29.6.1. Card identification 877
29.6.2. No data commands 879
29.6.3. Single block or multiple block write 879
29.6.4. Single block or multiple block read 880
29.6.5. Stream write and stream read (MMC only) 881
29.6.6. Erase 883
29.6.7. Bus width selection 884
29.6.8. Protection management 884
29.6.9. Card Lock/Unlock operation 885

29.7. Specific operations 887
29.7.1. SD I/O specific operations 887
29.7.2. CE-ATA specific operations 890
29.8. Register definition 892
29.8.1. Power control register (SDIO_PWRCTL) 892
29.8.2. Clock control register (SDIO_CLKCTL) 892
29.8.3. Command argument register (SDIO_CMDAGMT) 893
29.8.4. Command control register (SDIO_CMDCTL) 894
29.8.5. Command index response register (SDIO_RSPCMDIDX) 895
29.8.6. Response register (SDIO_RESPx) (x=0...3) 896
29.8.7. Data timeout register (SDIO_DATATO) 896
29.8.8. Data length register (SDIO_DATALEN) 897
29.8.9. Data control register (SDIO_DATACTL) 897
29.8.10. Data counter register (SDIO_DATACNT) 899
29.8.11. Status register (SDIO_STAT) 899
29.8.12. Interrupt clear register (SDIO_INTC) 901
29.8.13. Interrupt enable register (SDIO_INTEN) 902
29.8.14. FIFO counter register (SDIO_FIFOCNT) 904
29.8.15. FIFO data register (SDIO_FIFO) 904
30. External memory controller (EXMC) 906
30.1. Overview 906
30.2. Characteristics 906
30.3. Function overview 906
30.3.1. Block diagram 906
30.3.2. Basic regulation of EXMC access 907
30.3.3. External device address mapping 908
30.3.4. NOR/PSRAM controller 912
30.3.5. NAND flash or PC card controller 934
30.3.6. SDRAM controller 940
30.4. Register definition 952
30.4.1. NOR/PSRAM controller registers 952
30.4.2. NAND flash/PC card controller registers 957
30.4.3. SDRAM controller registers 963
30.4.4. SQPI-PSRAM controller registers 970
31. Controller area network (CAN) 974
31.1. Overview 974
31.2. Characteristics 974
31.3. Function overview 975
31.3.1. Working mode 975
31.3.2. Communication modes 976

31.3.3. Data transmission 977
31.3.4. Data reception 979
31.3.5. Filtering function 981
31.3.6. Time-triggered communication 984
31.3.7. Communication parameters 984
31.3.8. CAN FD operation 986
31.3.9. Transmitter Delay Compensation 987
31.3.10. Error flags 988
31.3.11. CAN interrupts 988
31.4. Register definition 991
31.4.1. Control register (CAN_CTL) 991
31.4.2. Status register (CAN_STAT) 992
31.4.3. Transmit status register (CAN_TSTAT) 994
31.4.4. Receive message FIFO0 register (CAN_RFIFO0) 996
31.4.5. Receive message FIFO1 register (CAN_RFIFO1) 997
31.4.6. Interrupt enable register (CAN_INTEN) 998
31.4.7. Error register (CAN_ERR) 999
31.4.8. Bit timing register (CAN_BT) 1000
31.4.9. FD control register (CAN_FDCTL) 1001
31.4.10. FD status register (CAN_FDSTAT) 1002
31.4.11. FD transmitter delay compensation register (CAN_FDTDC) 1003
31.4.12. Date Bit timing register (CAN_DBT) 1003
31.4.13. Transmit mailbox identifier register (CAN_TMIx) (x = 0...2) 1004
31.4.14. Transmit mailbox property register (CAN_TMPx) (x = 0…2) 1005
31.4.15. Transmit mailbox data0 register (CAN_TMDATA0x) (x = 0…2) 1006
31.4.16. Transmit mailbox data1 register (CAN_TMDATA1x) (x = 0...2) 1006
31.4.17. Rx FIFO mailbox identifier register (CAN_RFIFOMIx) (x = 0,1) 1007
31.4.18. Rx FIFO mailbox property register (CAN_RFIFOMPx) (x = 0,1) 1007
31.4.19. Rx FIFO mailbox data0 register (CAN_RFIFOMDATA0x) (x = 0,1) 1008
31.4.20. Rx FIFO mailbox data1 register (CAN_RFIFOMDATA1x) (x=0,1) 1009
31.4.21. Filter control register (CAN_FCTL) 1009
31.4.22. Filter mode configuration register (CAN_FMCFG) 1010
31.4.23. Filter scale configuration register (CAN_FSCFG) 1010
31.4.24. Filter associated FIFO register (CAN_FAFIFO) 1011
31.4.25. Filter working register (CAN_FW) 1011
31.4.26. Filter x data y register (CAN_FxDATAy) (x=0..27, y=0,1) 1012
32. Ethernet (ENET) 1013
32.1. Overview 1013
32.2. Characteristics 1013
32.2.1. Block diagram 1014
32.2.2. MAC 802.3 Ethernet packet description 1015
32.2.3. Ethernet signal description 1016

32.3. Function overview 1017
32.3.1. Interface configuration 1017
32.3.2. MAC function overview 1021
32.3.3. MAC statistics counters: MSC 1033
32.3.4. Wake up management: WUM 1033
32.3.5. Precision time protocol: PTP 1036
32.3.6. DMA controller description 1041
32.3.7. Example for a typical configuration flow of Ethernet 1065
32.3.8. Ethernet interrupts 1066
32.4. Register definition 1069
32.4.1. MAC configuration register (ENET_MAC_CFG) 1069
32.4.2. MAC frame filter register (ENET_MAC_FRMF) 1071
32.4.3. MAC hash list high register (ENET_MAC_HLH) 1073
32.4.4. MAC hash list low register (ENET_MAC_HLL) 1073
32.4.5. MAC PHY control register (ENET_MAC_PHY_CTL) 1074
32.4.6. MAC PHY data register (ENET_MAC_PHY_DATA) 1075
32.4.7. MAC flow control register (ENET_MAC_FCTL) 1075
32.4.8. MAC VLAN tag register (ENET_MAC_VLT) 1077
32.4.9. MAC remote wakeup frame filter register (ENET_MAC_RWFF) 1078
32.4.10. MAC wakeup management register (ENET_MAC_WUM) 1078
32.4.11. MAC debug register (ENET_MAC_DBG) 1079
32.4.12. MAC interrupt flag register (ENET_MAC_INTF) 1081
32.4.13. MAC interrupt mask register (ENET_MAC_INTMSK) 1082
32.4.14. MAC address 0 high register (ENET_MAC_ADDR0H) 1082
32.4.15. MAC address 0 low register (ENET_MAC_ADDR0L) 1083
32.4.16. MAC address 1 high register (ENET_MAC_ADDR1H) 1083
32.4.17. MAC address 1 low register (ENET_MAC_ADDR1L) 1084
32.4.18. MAC address 2 high register (ENET_MAC_ADDR2H) 1085
32.4.19. MAC address 2 low register (ENET_MAC_ADDR2L) 1085
32.4.20. MAC address 3 high register (ENET_MAC_ADDR3H) 1086
32.4.21. MAC address 3 low register (ENET_MAC_ADDR3L) 1087
32.4.22. MAC flow control threshold register (ENET_MAC_FCTH) 1087
32.4.23. MSC control register (ENET_MSC_CTL) 1088
32.4.24. MSC receive interrupt flag register (ENET_MSC_RINTF) 1089
32.4.25. MSC transmit interrupt flag register (ENET_MSC_TINTF) 1090
32.4.26. MSC receive interrupt mask register (ENET_MSC_RINTMSK) 1091
32.4.27. MSC transmit interrupt mask register (ENET_MSC_TINTMSK) 1092
32.4.28. MSC transmitted good frames after a single collision counter register (ENET_MSC_SCCNT) 1092
32.4.29. MSC transmitted good frames after more than a single collision counter register (ENET_MSC_MSCCNT) 1093
32.4.30. MSC transmitted good frames counter register (ENET_MSC_TGFCNT) 1093
32.4.31. MSC received frames with CRC error counter register (ENET_MSC_RFCECNT) 1094
32.4.32. MSC received frames with alignment error counter register (ENET_MSC_RFAECNT) 1094

32.4.33. MSC received good unicast frames counter register (ENET_MSC_RGUFCNT) 1095
32.4.34. PTP time stamp control register (ENET_PTP_TSCTL) 1095
32.4.35. PTP subsecond increment register (ENET_PTP_SSINC) 1098
32.4.36. PTP time stamp high register (ENET_PTP_TSH) 1098
32.4.37. PTP time stamp low register (ENET_PTP_TSL) 1098
32.4.38. PTP time stamp update high register (ENET_PTP_TSUH) 1099
32.4.39. PTP time stamp update low register (ENET_PTP_TSUL) 1099
32.4.40. PTP time stamp addend register (ENET_PTP_TSADDEND) 1100
32.4.41. PTP expected time high register (ENET_PTP_ETH) 1100
32.4.42. PTP expected time low register (ENET_PTP_ETL) 1101
32.4.43. PTP time stamp flag register (ENET_PTP_TSF) 1101
32.4.44. PTP PPS control register (ENET_PTP_PPSCTL) 1102
32.4.45. DMA bus control register (ENET_DMA_BCTL) 1102
32.4.46. DMA transmit poll enable register (ENET_DMA_TPEN) 1104
32.4.47. DMA receive poll enable register (ENET_DMA_RPEN) 1105
32.4.48. DMA receive descriptor table address register (ENET_DMA_RDTADDR) 1106
32.4.49. DMA transmit descriptor table address register (ENET_DMA_TDTADDR) 1106
32.4.50. DMA status register (ENET_DMA_STAT) 1107
32.4.51. DMA control register (ENET_DMA_CTL) 1110
32.4.52. DMA interrupt enable register (ENET_DMA_INTEN) 1113
32.4.53. DMA missed frame and buffer overflow counter register (ENET_DMA_MFBOCNT) ... 1115
32.4.54. DMA receive state watchdog counter register (ENET_DMA_RSWDC) 1115
32.4.55. DMA current transmit descriptor address register (ENET_DMA_CTDADDR) 1116
32.4.56. DMA current receive descriptor address register (ENET_DMA_CRDADDR) 1116
32.4.57. DMA current transmit buffer address register (ENET_DMA_CTBADDR) 1117
32.4.58. DMA current receive buffer address register (ENET_DMA_CRBADDR) 1117

33. Universal serial bus full-speed interface (USBFS) 1119

33.1. Overview 1119

33.2. Characteristics 1119

33.3. Block diagram 1120

33.4. Signal description 1120

33.5. Function overview 1120

33.5.1. USBFS clocks and working modes 1120
33.5.2. USB host function 1122
33.5.3. USB device function 1124
33.5.4. OTG function overview 1125
33.5.5. Data FIFO 1126
33.5.6. Operation guide 1129

33.6. Interrupts 1133

33.7. Register definition 1135

33.7.1. Global control and status registers 1135

33.7.2. Host control and status registers 1158
33.7.3. Device control and status registers 1169
33.7.4. Power and clock control register (USBFS_PWRCLKCTL) 1193
34. Universal serial bus high-speed interface (USBHS) 1194
34.1. Overview 1194
34.2. Characteristics 1194
34.3. Block diagram 1195
34.4. Signal description 1195
34.5. Function overview 1195
34.5.1. USBHS PHY selection, clocks and working modes 1195
34.5.2. USB host function 1198
34.5.3. USB device function 1201
34.5.4. OTG function overview 1202
34.5.5. Data FIFO 1203
34.5.6. DMA function 1206
34.5.7. Operation guide 1207
34.6. Interrupts 1213
34.7. Register definition 1215
34.7.1. USBHS global registers 1215
34.7.2. Host control and status registers 1239
34.7.3. Device control and status registers 1253
34.7.4. Power and clock control register (USBHS_PWRCLKCTL) 1282
35. Appendix 1283
35.1. List of abbreviations used in register 1283
35.2. List of terms 1283
35.3. Available peripherals 1284
36. Revision history 1285

List of Figures

Figure 1-1. The structure of the Cortex®-M33 processor 41
Figure 1-2. The system architecture of GD32F527xx devices 43
Figure 1-3. ECC decoder 48
Figure 2-1. GD32F527xx memory access architecture 71
Figure 2-2. Trusted code area protection 74
Figure 2-3. Trusted code area protection 75
Figure 2-4. Secure boot process 77
Figure 3-1. Process of page erase operation 87
Figure 3-2. Process of sector erase operation 88
Figure 3-3. Process of mass erase operation 89
Figure 3-4. Proccess of word program operation 91
Figure 4-1. Power supply overview 114
Figure 4-2. Waveform of the POR / PDR 116
Figure 4-3. Waveform of the BOR 117
Figure 4-4. Waveform of the LVD threshold 117
Figure 5-1. The system reset circuit 127
Figure 5-2. Clock tree 128
Figure 5-3. HXTAL clock source 130
Figure 5-4. HXTAL clock source in bypass mode 130
Figure 6-1. Block diagram of CTC 190
Figure 6-2. CTC trim counter 191
Figure 7-1. Block diagram of EXTI 203
Figure 8-1. Basic structure of a standard I / O 210
Figure 8-2. Basic structure of Input configuration 212
Figure 8-3. Basic structure of Output configuration 212
Figure 8-4. Basic structure of Analog configuration 213
Figure 8-5. Basic structure of Alternate function configuration 214
Figure 9-1. Block diagram of CRC calculation unit 228
Figure 10-1. TRNG block diagram 232
Figure 11-1. PKCAU module block diagram 238
Figure 11-2. Flow chart of RSA algorithm 239
Figure 11-3. Flow chart of ECDSA sign 240
Figure 11-4. Flow chart of ECDSA verification 241
Figure 11-5. Arithmetic addition 242
Figure 11-6. Arithmetic subtraction 243
Figure 11-7. Arithmetic multiplication 243
Figure 11-8. Arithmetic comparison 244
Figure 11-9. Modular reduction 244
Figure 11-10. Modular addition 245
Figure 11-11. Modular subtraction 245
Figure 11-12. Montgomery parameter calculation 246

Figure 11-13. Mutual mapping between Montgomery domain and natural domain 247
Figure 11-14. Montgomery multiplication 248
Figure 11-15. Modular exponentiation of normal mode 248
Figure 11-16. Modular exponentiation of fast mode 249
Figure 11-17. Modular inversion 249
Figure 11-18. RSA CRT exponentiation 250
Figure 11-19. Point on elliptic curve Fp check 252
Figure 11-20. ECC scalar multiplication of normal mode 253
Figure 11-21. ECC scalar multiplication of fast mode 253
Figure 11-22. ECDSA sign 255
Figure 11-23. ECDSA verification 256
Figure 12-1. DATAM No swapping and Half-word swapping 265
Figure 12-2. DATAM Byte swapping and Bit swapping 265
Figure 12-3. HAU block diagram 266
Figure 13-1. DATAM No swapping and Half-word swapping 279
Figure 13-2. DATAM Byte swapping and Bit swapping 280
Figure 13-3. CAU diagram 281
Figure 13-4. DES / TDES ECB encryption 282
Figure 13-5. DES / TDES ECB decryption 283
Figure 13-6. DES / TDES CBC encryption 284
Figure 13-7. DES / TDES CBC decryption 285
Figure 13-8. AES ECB encryption 286
Figure 13-9. AES ECB decryption 286
Figure 13-10. AES CBC encryption 287
Figure 13-11. AES CBC decryption 288
Figure 13-12. Counter block structure 288
Figure 13-13. AES CTR encryption / decryption 289
Figure 14-1. Block diagram of DMA 308
Figure 14-2. Data stream for three transfer modes 309
Figure 14-3. Handshake mechanism 310
Figure 14-4. Data packing/unpacking when PWIDTH = '00' 316
Figure 14-5. Data packing/unpacking when PWIDTH = '01' 316
Figure 14-6. Data packing/unpacking when PWIDTH = '10' 317
Figure 14-7. DMA operation of switch-buffer mode 318
Figure 14-8. System connection of DMA0 and DMA1 325
Figure 15-1. IPA block diagram 339
Figure 15-2. Pixel extension from 'RGB888' to 'ARGB8888' 343
Figure 15-3. Pixel extension from 'RGB565' to 'ARGB8888' 343
Figure 15-4. Pixel extension from 'ARGB1555' or 'ARGB4444' to 'ARGB8888' 343
Figure 15-5. Pixel compression 346
Figure 15-6. Inter timer operation 347
Figure 15-7. System connection of IPA 354
Figure 18-1. ADC module block diagram 387
Figure 18-2. Single operation mode 388

Figure 18-3. Continuous operation mode 389
Figure 18-4. Scan operation mode, continuous disable 390
Figure 18-5. Scan operation mode, continuous enable 390
Figure 18-6. Discontinuous operation mode 391
Figure 18-7. Data storage mode of 12-bit resolution 392
Figure 18-8. Data storage mode of 10-bit resolution 392
Figure 18-9. Data storage mode of 8-bit resolution 392
Figure 18-10. Data storage mode of 8-bit resolution 392
Figure 18-11. 20-bit to 16-bit result truncation 396
Figure 18-12. Numerical example with 5-bits shift and rounding 396
Figure 18-13. ADC sync block diagram 398
Figure 18-14. Routine parallel mode on 16 channels 399
Figure 18-15. Routine follow-up mode on 1 channel in continuous operation mode 400
Figure 19-1. DAC block diagram 417
Figure 19-2. DAC LFSR algorithm 419
Figure 19-3. DAC triangle noise wave 420
Figure 20-1. Free watchdog timer block diagram 433
Figure 20-2. Window watchdog timer block diagram 438
Figure 20-3. Window watchdog timing diagram 439
Figure 21-1. Block diagram of RTC 444
Figure 22-1. Advanced timer block diagram 477
Figure 22-2. Timing chart of internal clock divided by 1 478
Figure 22-3. Timing chart of PSC value change from 0 to 2 479
Figure 22-4. Timing chart of up counting mode, PSC=0/2 480
Figure 22-5. Timing chart of up counting mode, change TIMERx_CAR on the go 481
Figure 22-6. Timing chart of down counting mode, PSC=0/2 482
Figure 22-7. Timing chart of down counting mode, change TIMERx_CAR on the go 482
Figure 22-8. Center-aligned counter timechart 484
Figure 22-9. Repetition timechart for center-aligned counter 485
Figure 22-10. Repetition timechart for up-counter 485
Figure 22-11. Repetition timechart for down-counter 486
Figure 22-12. Channel input capture principle 487
Figure 22-13. Output-compare under three modes 489
Figure 22-14. EAPWM timechart 490
Figure 22-15. CAPWM timechart 490
Figure 22-16. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD) 492
Figure 22-17. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD) 492
Figure 22-18. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD) 492
Figure 22-19. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL 493
Figure 22-20. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD 493
Figure 22-21. Four Channels outputs in Composite PWM mode 494
Figure 22-22. Complementary output with dead-time insertion. 499
Figure 22-23. Output behavior in response to a break(The break high active) 500
Figure 22-24. Counter behavior with CI0FE0 polarity non-inverted in mode 2 501

Figure 22-25. Counter behavior with CI0FE0 polarity inverted in mode 2 501
Figure 22-26. Hall sensor is used to BLDC motor 502
Figure 22-27. Hall sensor timing between two timers 503
Figure 22-28. Restart mode 504
Figure 22-29. Pause mode 504
Figure 22-30. Event mode 505
Figure 22-31. Single pulse mode, TIMERx_CHxCV = 4, TIMERx_CAR=99 506
Figure 22-32. Timer0 master/slave mode timer example 506
Figure 22-33. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input 508
Figure 22-34. General Level 0 timer block diagram 542
Figure 22-35. Timing chart of internal clock divided by 1 543
Figure 22-36. Timing chart of PSC value change from 0 to 2 544
Figure 22-37. Timing chart of up counting mode, PSC=0/2 545
Figure 22-38. Timing chart of up counting mode, change TIMERx_CAR ongoing 545
Figure 22-39. Timing chart of down counting mode, PSC=0/2 546
Figure 22-40. Timing chart of down counting mode, change TIMERx_CAR ongoing 547
Figure 22-41. Timing chart of center-aligned counting mode 548
Figure 22-42. Channel input capture principle 549
Figure 22-43. Output-compare under three modes 551
Figure 22-44. EAPWM timechart 552
Figure 22-45. CAPWM timechart 552
Figure 22-46. Restart mode 554
Figure 22-47. Pause mode 554
Figure 22-48. Event mode 554
Figure 22-49. General level1 timer block diagram 583
Figure 22-50. Timing chart of internal clock divided by 1 584
Figure 22-51. Timing chart of PSC value change from 0 to 2 585
Figure 22-52. Timing chart of up counting mode, PSC=0/2 586
Figure 22-53. Timing chart of up counting mode, change TIMERx_CAR ongoing 586
Figure 22-54. Channel input capture principle 587
Figure 22-55. Output-compare under three modes 589
Figure 22-56. EAPWM timechart 590
Figure 22-57. CAPWM timechart 590
Figure 22-58. Restart mode 592
Figure 22-59. Pause mode 592
Figure 22-60. Event mode 593
Figure 22-61. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99 594
Figure 22-62. General level2 timer block diagram 609
Figure 22-63. Timing chart of internal clock divided by 1 609
Figure 22-64. Timing chart of PSC value change from 0 to 2 610
Figure 22-65. Timing chart of up counting mode, PSC=0/2 611
Figure 22-66. Timing chart of up counting mode, change TIMERx_CAR ongoing 611
Figure 22-67. Channel input capture principle 612
Figure 22-68. Output-compare under three modes 614

Figure 22-69. Basic timer block diagram 627
Figure 22-70. Timing chart of internal clock divided by 1 628
Figure 22-71. Timing chart of PSC value change from 0 to 2 628
Figure 22-72. Timing chart of up counting mode, PSC=0/2 629
Figure 22-73. Up-counter timechart, change TIMERx_CAR on the go 630
Figure 23-1. USART module block diagram 638
Figure 23-2. USART character frame (8 bits data and 1 stop bit) 638
Figure 23-3. USART transmit procedure 640
Figure 23-4. Receiving a frame bit by oversampling method (OSB=0) 641
Figure 23-5. Configuration step when use DMA for USART transmission 643
Figure 23-6. Configuration step when use DMA for USART reception 644
Figure 23-7. Hardware flow control between two USARTs 644
Figure 23-8. Hardware flow control 645
Figure 23-9. Break frame occurs during idle state 646
Figure 23-10. Break frame occurs during a frame 647
Figure 23-11. Example of USART in synchronous mode 647
Figure 23-12. 8-bit format USART synchronous waveform (CLEN=1) 648
Figure 23-13. IrDA SIR ENDEC module 648
Figure 23-14. IrDA data modulation 649
Figure 23-15. ISO7816-3 frame format 650
Figure 23-16. USART interrupt mapping diagram 652
Figure 24-1. I2C module block diagram 668
Figure 24-2. Data validation 669
Figure 24-3. START and STOP signal 669
Figure 24-4. Clock synchronization 670
Figure 24-5. SDA line arbitration 670
Figure 24-6. I2C communication flow with 7-bit address 671
Figure 24-7. I2C communication flow with 10-bit address (Master Transmit) 671
Figure 24-8. I2C communication flow with 10-bit address (Master Receive) 671
Figure 24-9. Programming model for slave transmitting (10-bit address mode) 673
Figure 24-10. Programming model for slave receiving (10-bit address mode) 674
Figure 24-11. Programming model for master transmitting (10-bit address mode) 675
Figure 24-12. Programming model for master receiving using Solution A (10-bit address mode) 677
Figure 24-13. Programming model for master receiving mode using solution B (10-bit address mode) 678
Figure 24-14. I2C module block diagram 697
Figure 24-15. Data validation 698
Figure 24-16. START and STOP signal 699
Figure 24-17. I2C communication flow with 10-bit address (Master Transmit) 699
Figure 24-18. I2C communication flow with 7-bit address (Master Transmit) 700
Figure 24-19. I2C communication flow with 7-bit address (Master Receive) 700
Figure 24-20. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0) .. 700
Figure 24-21. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1) .. 700
Figure 24-22. Data hold time 701

Figure 24-23. Data setup time 702
Figure 24-24. Data transmission 704
Figure 24-25. Data reception 704
Figure 24-26. I2C initialization in slave mode 707
Figure 24-27. Programming model for slave transmitting when SS=0 708
Figure 24-28. Programming model for slave transmitting when SS=1 709
Figure 24-29. Programming model for slave receiving 710
Figure 24-30. I2C initialization in master mode 711
Figure 24-31. Programming model for master transmitting (N<=255) 712
Figure 24-32. Programming model for master transmitting (N>255) 713
Figure 24-33. Programming model for master receiving (N<=255) 714
Figure 24-34. Programming model for master receiving (N>255) 715
Figure 24-35. SMBus master transmitter and slave receiver communication flow 718
Figure 24-36. SMBus master receiver and slave transmitter communication flow 719
Figure 25-1. Block diagram of SPI 737
Figure 25-2. SPI timing diagram in normal mode 738
Figure 25-3. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0) 739
Figure 25-4. A typical Full-duplex connection 741
Figure 25-5. A typical simplex connection (Master: Receive, Slave: Transmit) 742
Figure 25-6. A typical simplex connection (Master: Transmit only, Slave: Receive) 742
Figure 25-7. A typical bidirectional connection 742
Figure 25-8. Timing diagram of TI master mode with discontinuous transfer 744
Figure 25-9. Timing diagram of TI master mode with continuous transfer 744
Figure 25-10. Timing diagram of TI slave mode 745
Figure 25-11. Timing diagram of quad write operation in Quad-SPI mode 746
Figure 25-12. Timing diagram of quad read operation in Quad-SPI mode 747
Figure 25-13. Block diagram of I2S 750
Figure 25-14. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) 752
Figure 25-15. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) 752
Figure 25-16. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) 752
Figure 25-17. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) 752
Figure 25-18. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) 752
Figure 25-19. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) 753
Figure 25-20. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) 753
Figure 25-21. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) 753
Figure 25-22. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) 753
Figure 25-23. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) 754
Figure 25-24. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) 754
Figure 25-25. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) 754
Figure 25-26. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) 754
Figure 25-27. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) 754
Figure 25-28. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) 754
Figure 25-29. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) 755
Figure 25-30. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) 755

Figure 25-31. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) 755
Figure 25-32. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) 755
Figure 25-33. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) 756
Figure 25-34. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0) 756
Figure 25-35. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1) 756
Figure 25-36. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0) 756
Figure 25-37. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1) 756
Figure 25-38. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0) 757
Figure25-39. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1) 757
Figure 25-40. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0) 757
Figure 25-41. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1) 757
Figure 25-42. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0) 757
Figure25-43. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1) 758
Figure 25-44. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0) 758
Figure 25-45. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1) 758
Figure 25-46. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0) 758
Figure 25-47. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1) 758
Figure 25-48. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0) 759
Figure 25-49. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1) 759
Figure 25-50. Block diagram of I2S clock generator 759
Figure 26-1. block diagram 777
Figure 26-2. Clock divider logic 778
Figure 26-3 FS active width 781
Figure 26-4 FS polarity 781
Figure 26-5 FS function 782
Figure 26-6 Slot activation 783
Figure 26-7 Slot distribution when FUNC = 0 783
Figure 26-8 Slot distribution when FUNC = 1 783

Figure 26-9 Slot partition convention 784
Figure 26-10 Offset region treatment 784
Figure 26-11 SD output management 784
Figure 26-12 Data configuration 785
Figure 26-13 AC'97 slot partition 786
Figure 26-14 AC'97 tag definition 786
Figure 26-15 SPDIF data formate 788
Figure 26-16 Mute frame activation 790
Figure 26-17 Compander data path 791
Figure 26-18 Frame synchronization advanced detection 795
Figure 26-19 Frame synchronization postponed detection 796
Figure 27-1. DCI module block diagram 811
Figure 27-2. Hardware synchronization mode 812
Figure 27-3. Hardware synchronization mode: JPEG format supporting 813
Figure 28-1. TLI module block diagram 825
Figure 28-2. Display timing diagram 825
Figure 28-3. Block diagram of Blending 828
Figure 29-1. SDIO "no response" and "no data" operations 845
Figure 29-2. SDIO multiple blocks read operation 846
Figure 29-3. SDIO multiple blocks write operation 846
Figure 29-4. SDIO sequential read operation 846
Figure 29-5. SDIO sequential write operation 847
Figure 29-6. SDIO block diagram 847
Figure 29-7. Command Token Format 854
Figure 29-8. Response Token Format 866
Figure 29-9. 1-bit data bus width 869
Figure 29-10. 4-bit data bus width 869
Figure 29-11. 8-bit data bus width 870
Figure 29-12. Read wait control by stopping SDIO_CLK 888
Figure 29-13. Read wait operation using SDIO_DAT[2] 888
Figure 29-14. Function2 read cycle inserted during function1 multiple read cycle 889
Figure 29-15. Read Interrupt cycle timing 889
Figure 29-16. Write interrupt cycle timing 890
Figure 29-17. Multiple block 4-Bit read interrupt cycle timing 890
Figure 29-18. Multiple block 4-Bit write interrupt cycle timing 890
Figure 29-19. The operation for command completion disable signal 891
Figure 30-1. The EXMC block diagram 907
Figure 30-2. EXMC memory banks 908
Figure 30-3. Four regions of bank0 address mapping 909
Figure 30-4. NAND/PC card address mapping 909
Figure 30-5. Diagram of bank1 common space 910
Figure 30-6. SDRAM address mapping 911
Figure 30-7. Mode 1 read access 916
Figure 30-8. Mode 1 write access 916

Figure 30-9. Mode A read access 917
Figure 30-10. Mode A write access 918
Figure 30-11. Mode 2/B read access 919
Figure 30-12. Mode 2 write access 920
Figure 30-13. Mode B write access 920
Figure 30-14. Mode C read access 921
Figure 30-15. Mode C write access 922
Figure 30-16. Mode D read access 923
Figure 30-17. Mode D write access 924
Figure 30-18. Multiplex mode read access 925
Figure 30-19. Multiplex mode write access 926
Figure 30-20. Read access timing diagram under async-wait signal assertion 927
Figure 30-21. Write access timing diagram under async-wait signal assertion 928
Figure 30-22. Read timing of synchronous multiplexed burst mode 930
Figure 30-23. Write timing of synchronous multiplexed burst mode 931
Figure 30-24. SPI-PSRAM access 933
Figure 30-25. SQPI-PSRAM access 934
Figure 30-26. QPI-PSRAM access 934
Figure 30-27. Access timing of common memory space of NAND flash or PC card controller 936
Figure 30-28. Access to none "NCE don't care" NAND Flash 938
Figure 30-29. SDRAM controller block diagram 942
Figure 30-30. Burst read operation 945
Figure 30-31. Data sampling clock delay chain 946
Figure 30-32. Burst write operation 946
Figure 30-33. Read access when FIFO not hit (BRSTRD=1, CL=2, SDCLK=2, PIPED=2) 947
Figure 30-34. Read access when FIFO hit (BRSTRD=1) 948
Figure 30-35. Cross boundary read operation 949
Figure 30-36. Cross boundary write operation 949
Figure 30-37. Process for self-refresh entry and exit 950
Figure 30-38. Process for power-down entry and exit 950
Figure 31-1. CAN module block diagram 975
Figure 31-2. Transmission register 977
Figure 31-3. State of transmit mailbox 978
Figure 31-4. Reception register 980
Figure 31-5. 32-bit filter 981
Figure 31-6. 16-bit filter 981
Figure 31-7. 32-bit mask mode filter 981
Figure 31-8. 16-bit mask mode filter 982
Figure 31-9. 32-bit list mode filter 982
Figure 31-10. 16-bit list mode filter 982
Figure 31-11. The bit time 985
Figure 31-12. Transmitter Delay Measurement 988
Figure 32-1. ENET module block diagram 1014
Figure 32-2. MAC / Tagged MAC frame format 1015

Figure 32-3. Station management interface signals 1017
Figure 32-4. Media independent interface signals 1019
Figure 32-5. Reduced media-independent interface signals 1021
Figure 32-6. Wakeup frame filter register 1034
Figure 32-7. System time update using the fine correction method 1037
Figure 32-8. Descriptor ring and chain structure 1042
Figure 32-9. Transmit descriptor in normal mode 1047
Figure 32-10. Transmit descriptor in enhanced mode 1052
Figure 32-11. Receive descriptor in normal mode 1056
Figure 32-12. Receive descriptor in enhanced mode 1062
Figure 32-13. MAC interrupt scheme 1067
Figure 32-14. Ethernet interrupt scheme 1067
Figure 32-15. Wakeup frame filter register 1078
Figure 33-1. USBFS block diagram 1120
Figure 33-2. Connection with host or device mode 1121
Figure 33-3. Connection with OTG mode 1122
Figure 33-4. State transition diagram of host port 1122
Figure 33-5. HOST mode FIFO space in SRAM 1127
Figure 33-6. Host mode FIFO access register map 1127
Figure 33-7. Device mode FIFO space in SRAM 1128
Figure 33-8. Device mode FIFO access register map 1128
Figure 34-1. USBHS block diagram 1195
Figure 34-2. Connection using internal embedded PHY with host or device mode 1197
Figure 34-3. Connection using internal embedded PHY with OTG mode 1198
Figure 34-4. Connection using external ULPI PHY 1198
Figure 34-5. State transition diagram of host port 1199
Figure 34-6. HOST mode FIFO space in SRAM 1204
Figure 34-7. Host mode FIFO access register map 1204
Figure 34-8. Device mode FIFO space in SRAM 1205
Figure 34-9. Device mode FIFO access register map 1205

List of Tables

Table 1-1. Bus Interconnection Matrix 41
Table 1-2. Memory map of GD32F527xx devices 44
Table 1-3. Boot modes 50
Table 3-1. GD32F527xx base address and size for 4MB dual bank flash memory 82
Table 3-2. GD32F527xx base address and size for 2MB dual bank flash memory 83
Table 3-3. GD32F527xx base address and size for 1MB single bank flash memory 84
Table 3-4. GD32F527xx base address and size for 512KB single bank flash memory 84
Table 3-5. OTP0 lock 91
Table 3-6. OTP1 lock 92
Table 3-7. OTP2 lock 92
Table 3-8. Option byte 93
Table 3-9. WP0/WP1 bit for sectors protected 96
Table 3-10. Security protection 97
Table 3-11. system parameters 97
Table 4-1. Power saving mode summary 120
Table 5-1. Clock output 0 source select 133
Table 5-2. Clock output 1 source select 133
Table 5-3. 1.2V domain voltage selected in deep-sleep mode 134
Table 7-1. NVIC exception types in Cortex®-M33 200
Table 7-2. Interrupt vector table 200
Table 7-3. EXTI source 204
Table 8-1. GPIO configuration table 210
Table 11-1. Parameters of RSA algorithm 239
Table 11-2. Integer arithmetic operations 241
Table 11-3. Range of parameters used by RSA CRT exponentiation operation 251
Table 11-4. Elliptic curve operations in Fp domain 251
Table 11-5. Range of parameters used by point on elliptic curve Fp check 252
Table 11-6. Range of parameters used by ECC scalar multiplication 254
Table 11-7. Range of parameters used by ECDSA sign 255
Table 11-8. Range of parameters used by ECDSA verification 257
Table 11-9. Modular exponentiation computation times 258
Table 11-10. ECC scalar multiplication computation times 259
Table 11-11. ECDSA signature average computation times 259
Table 11-12. ECDSA verification average computation times 259
Table 11-13. Montgomery parameters average computation times 259
Table 11-14. PKCAU interrupt requests 260
Table 14-1. Transfer mode 309
Table 14-2. Peripheral requests to DMA0 310
Table 14-3. Peripheral requests to DMA1 311
Table 14-4. CNT configuration 313
Table 14-5. FIFO counter critical value configuration rules 314

Table 14-6. DMA interrupt events 323
Table 15-1. IPA conversion mode 340
Table 15-2. Foreground and background CLUT pixel format 341
Table 15-3. Foreground and background pixel format 342
Table 15-4. Alpha channel value modulation 344
Table 15-5. Destination pixel format 345
Table 15-6. IPA interrupt events 352
Table 16-1. Pin assignment 375
Table 18-1. ADC internal input signals 386
Table 18-2. ADC input pins definition 386
Table 18-3. External trigger modes 393
Table 18-4. External trigger source for ADC 393
Table 18-5. t_{CONV} timings depending on resolution 395
Table 18-6. Maximum output results vs N and M Grayed values indicates truncation 397
Table 18-7. ADC sync mode table 397
Table 19-1. DAC I/O description 417
Table 19-2. DAC triggers and outputs summary 417
Table 19-3. Triggers of DAC 418
Table 20-1. Min/max FWDGT timeout period at 32 kHz (IRC32K) 433
Table 20-2. Min / max timeout value at 50 MHz (f_{PCLK1}) 439
Table 21-1 RTC power saving mode management 455
Table 21-2 RTC interrupts control 455
Table 22-1. Timers (TIMERx) are divided into five sorts 475
Table 22-2.The Composite PWM pulse width 491
Table 22-3. Complementary outputs controlled by parameters 496
Table 22-4. Counting direction in different quadrature decoder mode 501
Table 22-5. Slave mode examples 503
Table 22-6. Slave mode examples 553
Table 22-7. Slave mode examples 591
Table 23-1. Description of USART important pins 637
Table 23-2. Configuration of stop bits 638
Table 23-3. USART interrupt requests 652
Table 24-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors) 668
Table 24-2. Event status flags 683
Table 24-3. Error flags 683
Table 24-4. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors) 697
Table 24-5. Data setup time and data hold time 703
Table 24-6. Communication modes to be shut down 704
Table 24-7. I2C error flags 720
Table 24-8. I2C interrupt events 720
Table 25-1. SPI signal description 737
Table 25-2. Quad-SPI signal description 738

Table 25-3. NSS function in slave mod 739
Table 25-4. NSS function in master mod 740
Table 25-5. SPI operation modes 740
Table 25-6. SPI interrupt requests 750
Table 25-7. I2S bitrate calculation formulas 759
Table 25-8. Audio sampling frequency calculation formulas 760
Table 25-9. Direction of I2S interface signals for each operation mode 760
Table 25-10. I2S interrupt 765
Table 26-1. Commonly used audio sampling rate 779
Table 26-2 FIFO request generation conditions 785
Table 26-3 AC'97 transmitter tag definition 787
Table 26-4 AC'97 receiver tag definition 787
Table 26-5. SOPD mode 788
Table 26-6 Parity bit calculation 788
Table 26-7 Mute frame composition 790
Table 26-8 A-law encoding 791
Table 26-9 A-law decoding 791
Table 26-10 Mu-law encoding 792
Table 26-11 Mu-law decoding 793
Table 26-12 Interrupt control 797
Table 27-1. PINs used by DCI 812
Table 27-2. Memory view in byte padding mode 814
Table 27-3. Memory view in half-word padding mode 815
Table 27-4. Status/Error flags 815
Table 28-1. Pins of display interface provided by TLI 825
Table 28-2. Supported pixel formats 827
Table 28-3. Status flags 829
Table 28-4. Error flags 829
Table 29-1. SDIO I/O definitions 848
Table 29-2. Command format 854
Table 29-3. Card command classes (CCCs) 855
Table 29-4. Basic commands (class 0) 857
Table 29-5. Block-Oriented read commands (class 2) 858
Table 29-6. Stream read commands (class 1) and stream write commands (class 3) 859
Table 29-7. Block-Oriented write commands (class 4) 860
Table 29-8. Erase commands (class 5) 861
Table 29-9. Block oriented write protection commands (class 6) 861
Table 29-10. Lock card (class 7) 862
Table 29-11. Application-specific commands (class 8) 862
Table 29-12. I/O mode commands (class 9) 863
Table 29-13. Switch function commands (class 10) 865
Table 29-14. Response R1 866
Table 29-15. Response R2 866
Table 29-16. Response R3 867

Table 29-17. Response R4 for MMC 867
Table 29-18. Response R4 for SD I/O 867
Table 29-19. Response R5 for MMC 868
Table 29-20. Response R5 for SD I/O 868
Table 29-21. Response R6 868
Table 29-22. Response R7 869
Table 29-23. Card status 871
Table 29-24. SD status 873
Table 29-25. Performance move field 875
Table 29-26. AU_SIZE field 875
Table 29-27. Maximum AU size 876
Table 29-28. Erase size field 876
Table 29-29. Erase timeout field 877
Table 29-30. Erase offset field 877
Table 29-31. Lock card data structure 885
Table 29-32. SDIO_RESPx register at different response type 896
Table 30-1. SDRAM mapping 911
Table 30-2. NOR flash interface signals description 912
Table 30-3. PSRAM non-muxed signal description 913
Table 30-4. SQPI-PSRAM signal description 913
Table 30-5. EXMC bank 0 supports all transactions 913
Table 30-6. NOR / PSRAM controller timing parameters 914
Table 30-7. EXMC_timing models 915
Table 30-8. Mode 1 related registers configuration 916
Table 30-9. Mode A related registers configuration 918
Table 30-10. Mode 2/B related registers configuration 920
Table 30-11. Mode C related registers configuration 922
Table 30-12. Mode D related registers configuration 924
Table 30-13. Multiplex mode related registers configuration 926
Table 30-14. Timing configurations of synchronous multiplexed read mode 930
Table 30-15. Timing configurations of synchronous multiplexed write mode 931
Table 30-16. SPI/QPI interface 932
Table 30-17. 8-bit or 16-bit NAND interface signal 935
Table 30-18. 16-bit PC card interface signal 935
Table 30-19. Bank1/2/3 of EXMC support the memory and access mode 935
Table 30-20. NAND flash or PC card programmable parameters 936
Table 30-21. SDRAM command truth table 942
Table 30-22. IO definition of SDRAM controller 943
Table 31-1. 32-bit filter number 982
Table 31-2. Filtering index 983
Table 31-3. CAN Event / Interrupt flags 990
Table 32-1. Ethernet pin configuration 1016
Table 32-2. Clock range 1018
Table 32-3. Rx interface signal encoding 1020

Table 32-4. Destination address filtering table 1029
Table 32-5. Source address filtering table 1029
Table 32-6. Error status decoding in RDES0, only used for normal descriptor (DFM=0) 1060
Table 32-7. Supported time stamp snapshot with PTP register configuration 1097
Table 33-1. USBFS signal description 1120
Table 33-2. USBFS global interrupt 1133
Table 34-1. USBHS signal description 1195
Table 34-2. USBHS supported speeds 1196
Table 34-3. USBHS global interrupt 1213
Table 35-1. List of abbreviations used in register 1283
Table 35-2. List of terms 1283
Table 36-1. Revision history 1285

1. System and memory architecture

The GD32F527xx series are 32-bit general-purpose microcontrollers based on the Arm® Cortex®-M33 processor. The Arm® Cortex®-M33 processor includes two AHB buses known as Code and System buses. All memory accesses of the Arm® Cortex®-M33 processor are executed on these two buses according to the different purposes and the target memory spaces. 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.

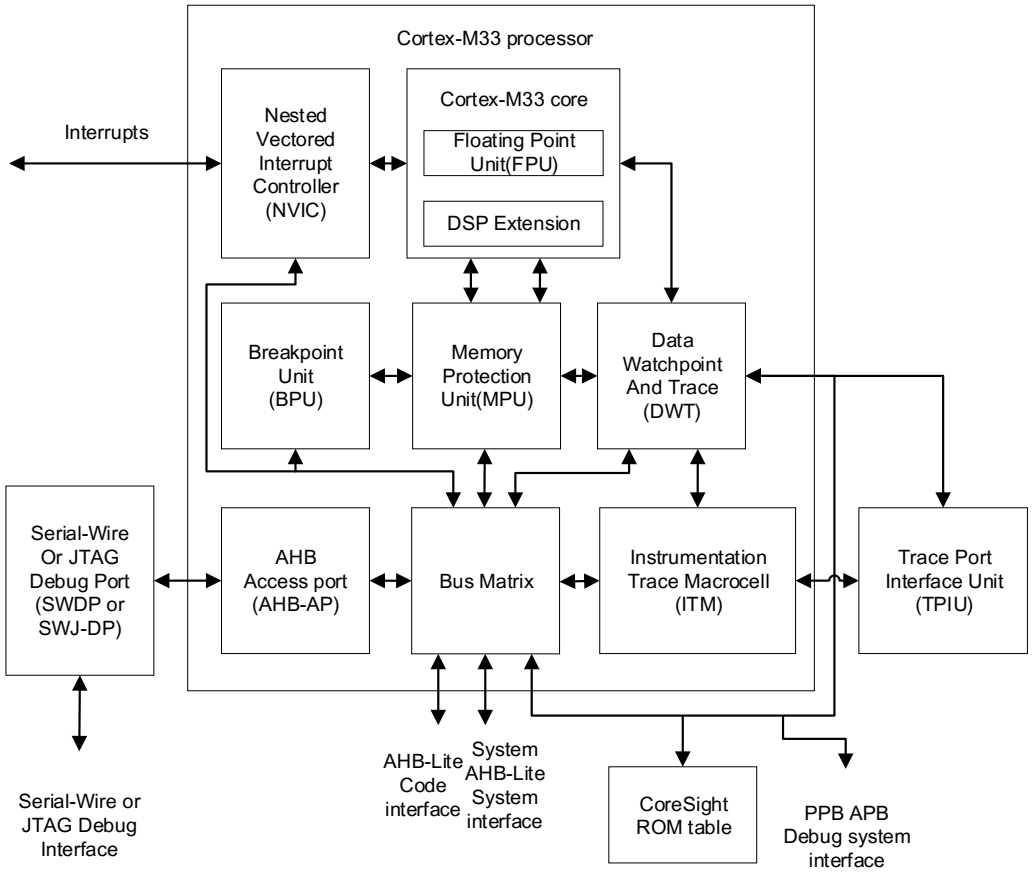
1.1. Arm® Cortex®-M33 processor

The Cortex®-M33 processor is a 32-bit processor that possesses low interrupt latency and low-cost debug. The characteristics of integrated and advanced make the Cortex®-M33 processor suitable for market products that require microcontrollers with high performance and low power consumption. The Cortex®-M33 processor is based on the Armv8 architecture and supports a powerful and scalable instruction set including general data processing I/O control tasks, advanced data processing bit field manipulations and DSP. Some system peripherals listed below are also provided by Cortex®-M33:

- Internal Bus Matrix connected with Code bus, System bus, and Private Peripheral Bus (PPB) and debug accesses
- Nested Vectored Interrupt Controller (NVIC)
- Breakpoint Unit (BPU)
- Data Watchpoint and Trace (DWT)
- Instrumentation Trace Macrocell (ITM)
- Serial Wire JTAG Debug Port (SWJ-DP)
- Trace Port Interface Unit (TPIU)
- Memory Protection Unit (MPU)
- Floating Point Unit (FPU)
- DSP Extension (DSP)

The following figure ***Figure 1-1. The structure of the Cortex®-M33 processor*** shows the Arm® Cortex®-M33 processor block diagram. For more information, refer to the Arm® Cortex®-M33 Technical Reference Manual.

Figure 1-1. The structure of the Cortex®-M33 processor



1.2. System architecture

A 32-bit multilayer bus is implemented in the GD32F527xx devices, which enables parallel access paths between multiple masters and slaves in the system. The multilayer bus consists of an AHB interconnect matrix, one AHB bus and two APB buses. The interconnection relationship of the AHB interconnect matrix is shown below. In the following table, "1" indicates the corresponding master is able to access the corresponding slave through the AHB interconnect matrix, while the blank means the corresponding master cannot access the corresponding slave through the AHB interconnect matrix.

This architecture is shown in ***Table 1-1. Bus Interconnection Matrix***.

Table 1-1. Bus Interconnection Matrix

	CBUS	SBUS	DMA0M	DMA0P	DMA1M	DMA1P	ENET	TLI	USBHS	IPA
FMC	1		1		1	1	1	1	1	1
TCMSRAM	1									
SRAM0	1	1	1		1	1	1	1	1	1

	CBUS	SBUS	DMA0M	DMA0P	DMA1M	DMA1P	ENET	TLI	USBHS	IPA
SRAM1		1	1		1	1	1	1	1	1
SRAM2		1	1		1	1	1	1	1	1
ADDSRAM	1	1	1		1	1	1	1	1	1
EXMC	1	1	1		1	1	1	1	1	1
AHB1		1		1		1				
AHB2		1			1	1				
APB1		1		1	1	1				
APB2		1		1	1	1				

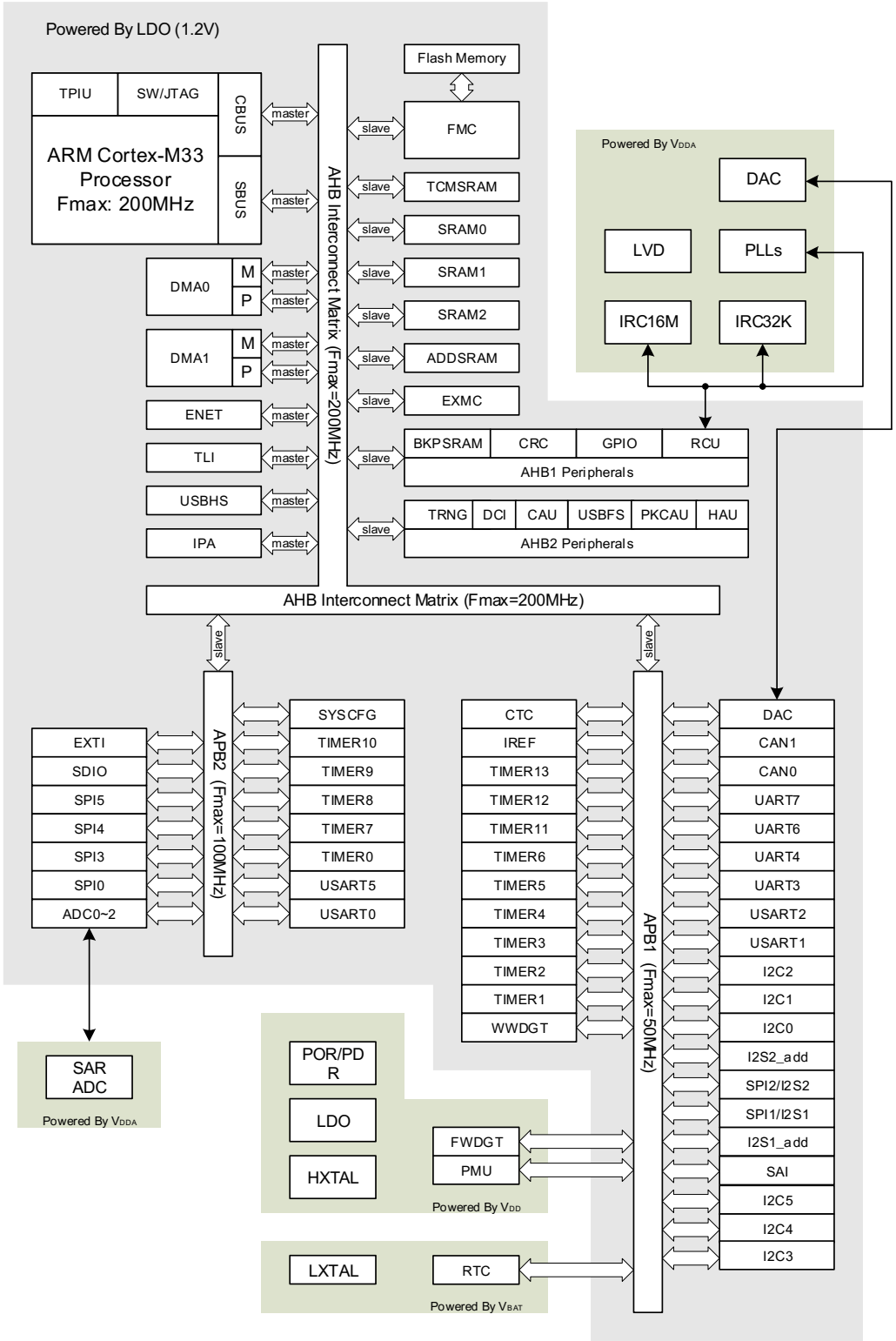
As is shown above, there are ten masters connected with the AHB interconnect matrix, including CBUS, SBUS, DMA0M, DMA0P, DMA1M, DMA1P, ENET, TLI, USBHS and IPA. CBUS is the code bus of the Cortex®-M33 core, which is used for any instruction fetch and data access to the Code region. Similarly, SBUS is the system bus of the Cortex®-M33 core, which is used for instruction/vector fetches, data loading/storing and debugging access of the system regions. The System regions include the internal SRAM region and the Peripheral region. DMA0M and DMA1M are the memory buses of DMA0 and DMA1 respectively. DMA0P and DMA1P are the peripheral buses of DMA0 and DMA1 respectively. ENET is the Ethernet. TLI is the TFT LCD interface. USBHS is the high-speed USB. And IPA is the image processing accelerator.

There are also eleven slaves connected with the AHB interconnect matrix, including FMC, TCMSRAM, SRAM0, SRAM1, SRAM2, ADDSRAM, EXMC, AHB1, AHB2, APB1 and APB2. FMC is the flash memory controller. TCMSRAM is the tightly-coupled memory SRAM. SRAM0, SRAM1 and SRAM2 are on-chip static random access memories. ADDSRAM is the additional SRAM, which is available only in some particular GD32F527xx devices. EXMC is the external memory controller. AHB1 and AHB2 are the two AHB buses connected with all of the AHB slaves, while APB1 and APB2 are the two APB buses connected with all of the APB slaves.

The two APB buses connect with all the APB peripherals. APB1 is up to 50MHz, APB2 operates up to 100MHz depending on the device.

The system architecture of GD32F527xx devices is shown in the ***Figure 1-2. The system architecture of GD32F527xx devices***.

Figure 1-2. The system architecture of GD32F527xx devices



1.3. Memory map

The Arm® Cortex®-M33 processor is structured in Harvard architecture which can use

separate buses to fetch instructions and load/store data. Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space which is the maximum address range of the Cortex®-M33 since the bus address width is 32-bit. Additionally, a pre-defined memory map is provided by the Cortex®-M33 processor to reduce the software complexity of repeated implementation of different device vendors. In the map, some regions are used by the Arm® Cortex®-M33 system peripherals which can not be modified. However, the other regions are available to the vendors. ***Table 1-2. Memory map of GD32F527xx devices*** shows the memory map of the GD32F527xx 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 GD32F527xx devices

Pre-defined Regions	Bus	Address	Peripherals
External Device	AHB matrix	0xC000 0000 - 0xDFFF FFFF	EXMC - SDRAM
		0xA000 1000 - 0xBFFF FFFF	Reserved
		0xA000 0000 - 0xA000 0FFF	EXMC - SWREG
External RAM		0x9000 0000 - 0x9FFF FFFF	EXMC - PC CARD
		0x7000 0000 - 0x8FFF FFFF	EXMC - NAND
		0x6000 0000 - 0x6FFF FFFF	EXMC - NOR / PSRAM / SRAM
Peripheral	AHB2	0x5006 3000 - 0x5FFF FFFF	Reserved
		0x5006 1000 - 0x5006 2FFF	PKCAU
		0x5006 0C00 - 0x5006 0FFF	Reserved
		0x5006 0800 - 0x5006 0BFF	TRNG
		0x5006 0400 - 0x5006 07FF	HAU
		0x5006 0000 - 0x5006 03FF	CAU
		0x5005 0400 - 0x5005 FFFF	Reserved
		0x5005 0000 - 0x5005 03FF	DCI
		0x5004 0000 - 0x5004 FFFF	Reserved
		0x5000 0000 - 0x5003 FFFF	USBFS
	AHB1	0x4008 0000 - 0x4FFF FFFF	Reserved (AHB1)
		0x4004 0000 - 0x4007 FFFF	USBHS
		0x4002 BC00 - 0x4003 FFFF	Reserved
		0x4002 B000 - 0x4002 BBFF	IPA
		0x4002 A000 - 0x4002 AFFF	Reserved
		0x4002 8000 - 0x4002 9FFF	ENET
		0x4002 6800 - 0x4002 7FFF	Reserved
		0x4002 6400 - 0x4002 67FF	DMA1
		0x4002 6000 - 0x4002 63FF	DMA0
		0x4002 5000 - 0x4002 5FFF	Reserved
		0x4002 4000 - 0x4002 4FFF	BKPSRAM
		0x4002 3C00 - 0x4002 3FFF	FMC

Pre-defined Regions	Bus	Address	Peripherals
		0x4002 3800 - 0x4002 3BFF	RCU
		0x4002 3400 - 0x4002 37FF	Reserved
		0x4002 3000 - 0x4002 33FF	CRC
		0x4002 2C00 - 0x4002 2FFF	Reserved
		0x4002 2800 - 0x4002 2BFF	Reserved
		0x4002 2400 - 0x4002 27FF	Reserved
		0x4002 2000 - 0x4002 23FF	GPIOI
		0x4002 1C00 - 0x4002 1FFF	GPIOH
		0x4002 1800 - 0x4002 1BFF	GPIOG
		0x4002 1400 - 0x4002 17FF	GPIOF
		0x4002 1000 - 0x4002 13FF	GPIOE
		0x4002 0C00 - 0x4002 0FFF	GPIOD
		0x4002 0800 - 0x4002 0BFF	GPIOC
		0x4002 0400 - 0x4002 07FF	GPIOB
		0x4002 0000 - 0x4002 03FF	GPIOA
	APB2	0x4001 8400 - 0x4001 FFFF	Reserved
		0x4001 8000 - 0x4001 83FF	Reserved
		0x4001 7C00 - 0x4001 7FFF	Reserved
		0x4001 7800 - 0x4001 7BFF	Reserved
		0x4001 7400 - 0x4001 77FF	Reserved
		0x4001 7000 - 0x4001 73FF	Reserved
		0x4001 6C00 - 0x4001 6FFF	Reserved
		0x4001 6800 - 0x4001 6BFF	TLI
		0x4001 5C00 - 0x4001 67FF	Reserved
		0x4001 5800 - 0x4001 5BFF	SAI
		0x4001 5400 - 0x4001 57FF	SPI5
		0x4001 5000 - 0x4001 53FF	SPI4
		0x4001 4C00 - 0x4001 4FFF	Reserved
		0x4001 4800 - 0x4001 4BFF	TIMER10
		0x4001 4400 - 0x4001 47FF	TIMER9
		0x4001 4000 - 0x4001 43FF	TIMER8
		0x4001 3C00 - 0x4001 3FFF	EXTI
		0x4001 3800 - 0x4001 3BFF	SYSCFG
		0x4001 3400 - 0x4001 37FF	SPI3
		0x4001 3000 - 0x4001 33FF	SPI0
		0x4001 2C00 - 0x4001 2FFF	SDIO
		0x4001 2800 - 0x4001 2BFF	Reserved
		0x4001 2400 - 0x4001 27FF	Reserved
		0x4001 2000 - 0x4001 23FF	ADC

Pre-defined Regions	Bus	Address	Peripherals
		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 C800 - 0x4000 FFFF	Reserved
		0x4000 C400 - 0x4000 C7FF	IREF
		0x4000 C000 - 0x4000 C3FF	Reserved
		0x4000 9000 - 0x4000 BFFF	Reserved
		0x4000 8C00 - 0x4000 8FFF	CAN SRAM 1k-2k bytes
		0x4000 8800 - 0x4000 8BFF	I2C5
		0x4000 8400 - 0x4000 87FF	I2C4
		0x4000 8000 - 0x4000 83FF	I2C3
		0x4000 7C00 - 0x4000 7FFF	UART7
		0x4000 7800 - 0x4000 7BFF	UART6
		0x4000 7400 - 0x4000 77FF	DAC
		0x4000 7000 - 0x4000 73FF	PMU
		0x4000 6C00 - 0x4000 6FFF	CTC
		0x4000 6800 - 0x4000 6BFF	CAN1
		0x4000 6400 - 0x4000 67FF	CAN0
		0x4000 6000 - 0x4000 63FF	CAN SRAM 0-1k bytes
		0x4000 5C00 - 0x4000 5FFF	I2C2
		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	I2S2_add
		0x4000 3C00 - 0x4000 3FFF	SPI2/I2S2
		0x4000 3800 - 0x4000 3BFF	SPI1/I2S1
		0x4000 3400 - 0x4000 37FF	I2S1_add
		0x4000 3000 - 0x4000 33FF	FWDGT
		0x4000 2C00 - 0x4000 2FFF	WWDGT
		0x4000 2800 - 0x4000 2BFF	RTC
		0x4000 2400 - 0x4000 27FF	Reserved

Pre-defined Regions	Bus	Address	Peripherals
		0x4000 2000 - 0x4000 23FF	TIMER13
		0x4000 1C00 - 0x4000 1FFF	TIMER12
		0x4000 1800 - 0x4000 1BFF	TIMER11
		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		0x2010 0000 - 0x3FFF FFFF	Reserved
		0x2008 0000 - 0x200F FFFF	ADDSRAM (512KB)
		0x2005 0000 - 0x2007 FFFF	SRAM2(192KB)
		0x2004 0000 - 0x2004 FFFF	SRAM1(64KB)
		0x2000 0000 - 0x2003 FFFF	SRAM0(256KB)
Code		0x1FFF C010 - 0x1FFF FFFF	Reserved
		0x1FFF C000 - 0x1FFF C00F	Option bytes Block (Bank0 option)
		0x1FFF B000 - 0x1FFF BFFF	Reserved
		0x1FFF 7880 - 0x1FFF AFFF	Reserved
		0x1FFF 7840 - 0x1FFF 787F	OTP0 Block (lock)
		0x1FFF 7800 - 0x1FFF 783F	OTP0 Block (data)
		0x1FFF 0000 - 0x1FFF 77FF	Boot loader(30KB)
		0x1FFE C010 - 0x1FFE FFFF	Reserved
		0x1FFE C000 - 0x1FFE C00F	Option bytes Block (Bank1 option)
		0x1FF2 0230 - 0x1FFE BFFF	Reserved
		0x1FF2 0210 - 0x1FF2 022F	OTP Block2 (lock)
		0x1FF2 0200 - 0x1FF2 020F	OTP Block1 (lock)
		0x1FF2 0000 - 0x1FF2 01FF	OTP Block2 (data)
		0x1FF0 0000 - 0x1FF1 FFFF	OTP Block1 (data)
		0x1001 0000 - 0x1FEF FFFF	Reserved
		0x1000 0000 - 0x1000 FFFF	TCMSRAM (64KB)
		0x0880 0000 - 0x0FFF FFFF	Reserved
		0x0840 0000 - 0x0877 FFFF	Main Flash(Bank1_Ex 3584kB)
		0x0820 0000 - 0x083F FFFF	Main Flash(Bank1 2MB)
		0x0800 0000 - 0x081F FFFF	Main Flash(Bank0 2MB)
		0x0030 0000 - 0x07FF FFFF	Aliased to the boot device
		0x0010 0000 - 0x002F FFFF	
		0x0002 0000 - 0x000F FFFF	
		0x0000 0000 - 0x0001 FFFF	

1.3.1. On-chip SRAM memory

The GD32F527xx series contain up to 512KB of on-chip SRAM which starts at the address 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses.

ECC

When reading and writing SRAM, it supports 7-bit ECC function. It can correct 1 bit error and detect multiple bits (two bits) error.

It must be written before reading SRAM, otherwise it may cause ECC error. Unaligned read operations will be performed in accordance with 32-bit read operations. Non-aligned write operations will produce a read-modify-write process. For example, when 16-bit data is written into SRAM, firstly the another 16-bit data is read out from the SRAM, and the 16-bit that need to be written are combined to a 32-bit data, and finally the 32-bit data is written into the SRAM together. Therefore, when initializing SRAM, it can only be written in a 32-bit width.

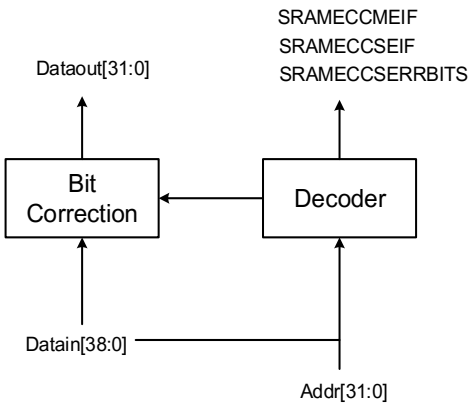
The ECC module is composed of an encoder and a decoder.

Encoder: When performing a SRAM write operation, a 7-bit ECC code will be generated and written into the SRAM together with the data.

Decoder: When performing a SRAM read operation, it uses the same algorithm as the encoder to decode and generate a 7-bit ECC code. The ECC code includes ECC error status and information which specific bit of the 32-bit data has single bit error.

The decoder is shown in the figure *Figure 1-3. ECC decoder* below:

Figure 1-3. ECC decoder



EEIC

The EEIC (ECC Error Interrupt Control) module provides the function of ECC error status management and ECC interrupt configuration.

Enable ECCEN in Option byte and then By setting the SYSCFG_STAT register and SYSCFG_SRAM0ECC,SYSCFG_SRAM1ECC,SYSCFG_SRAM2ECC,SYSCFG_ADDSRA

MECC, SYSCFG_TCMSRAMECC, SYSCFG_BKPSRAMECC register can realize the ECC error detection of SRAM0, SRAM1, SRAM2, ADDSRAM and TCMSRAM respectively. Taking SRAM0 as an example. The ECC configuration bit of SRAM1, SRAM2, ADDSRAM, TCMSRAM , FLASH is similar to SRAM0.

Single bit error correction event

When a single-bit error correction event is detected in SRAM，EEIC：

(1) The ECCSEIF0 bit in SYSCFG_STAT register will be set. Software can clear it by writing 1.
(2) The ECCADDR0 bit in SYSCFG_SRAM0ECC records the address where the single-bit error correction event occurred.

Two bits non-correction error event

When a two bits non-correction error event is detected in SRAM，EEIC：

(1) The ECCMEIF0 bit in SYSCFG_STAT register will be set. Software can clear it by writing 1.
(2) The ECCADDR0 bit in SYSCFG_SRAM0ECC records the address where the two bits non-correction error event occurred.

Single bit error correction interrupt

Set the ECCSEIE0 bit in SYSCFG_SRAM0ECC register. When a single-bit error correctable event is detected, a corresponding interrupt will be generated.

Two bits non-correction error interrupt

Set the ECCSEIE0 bit in SYSCFG_SRAM0ECC register. When a two bits error non-correction event is detected, a NMI interrupt will be generated.

1.3.2. On-chip Flash memory

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

- Up to 7680KB of main Flash memory (include Extension Main Flash).
- Up to 30KB of information blocks for the boot loader.
- Option bytes to configure the device.

Refer to ***Flash memory controller (FMC)*** for more details.

1.4. Boot configuration

GD32F527xx devices provide for kinds of boot sources which can be selected by Security Protection, EFUSE, BOOT0 and BOOT1 pins. The details are shown in the following table ***Table 1-3. Boot modes***.The value on the two pins is latched on the 4th rising edge of CK_SYS after a reset. It is up to the user to set the BOOT0 and BOOT1 pins after a power-on reset or a system reset to select the required boot source. Once the two pins have been

sampled, they are free and can be used for other purposes.

Table 1-3. Boot modes

Security Protection	EFUSE		Boot pad		BOOT_MODE[2:0]	Boot Select
	NBTSB	BTFOSEL	BOOT0	BOOT1		
no protection/ Protection level low	0	x	1	1	011	SRAM
no protection/ Protection level low	0	x	1	0	001	BootLoader
no protection/ Protection level low	0	0	0	x	000	Main Flash
no protection/ Protection level low	0	1	0	x	101	OTP1
x	1	0	x	x	000	Main Flash
x	1	1	x	x	101	OTP1
Protection level high	x	0	x	x	000	Main Flash
Protection level high	x	1	x	x	101	OTP1

After power-on sequence or a system reset, the Arm® Cortex®-M33 processor fetches the top-of-stack value from address 0x0000 0000 and the base address of boot code from 0x0000 0004 in sequence. Then, it starts executing code from the base address of boot code.

The corresponding memory space of the selected boot source is aliased in the boot memory space which begins at the address 0x0000 0000. When the on-chip SRAM is selected as the boot source, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and the offset register.

By configure Security Protection and EFUSE, can to select the BOOT mode as shown in ***Table 1-3. Boot modes***

When the security protection level and the NBTSB and BTFOSEL bits in EFUSE_CTL register are set, configure BOOT0, BOOT1 pin values to select Boot from SRAM, BootLoader, Main Flash, OTP. The values of the BOOT0 and BOOT1 pins can be decided by the configuration of BOOT_MODE[2:0] in the SYSCFG_CFG0 register (Note: Once these control bits are written through software, the level states of the BOOT0 and BOOT1 pins will be ignored).

When the main flash memory is selected to be the boot source, the memory space beginning at the address 0x0800 0000 is aliased in the boot memory space. Since either Bank 0 or Bank 1 of the main flash memory can be mapped at address 0x0800 0000 according to the

configuration of the FMC_SWP bit in the register SYSCFG_CFG0 (refer to *Configuration register 0 (SYSCFG_CFG0)* for more details), the device can either boot from Bank 0 or from Bank 1. When OTP is selected to be the boot source, the memory space beginning at the address 0x1FF0 0000 is aliased in the boot memory space.

In order to enable boot bank function, the BB bit in the option bytes has to be set. When this bit is set and the main flash memory is selected as the boot source, the device can boot from the boot loader, and the boot loader jumps to execute the code in Bank 1 of the main flash memory.

Note: To ensure system reliability and stability, during the hardware design phase, refer to the Boot mode section of the datasheet for the corresponding model. Confirm that peripherals which may generate interference signals (such as sensor outputs, watchdog chips, keys, etc.) are not connected to the pins used by the Bootloader, so as to avoid affecting the normal function of the Bootloader and causing abnormal outputs from the connected peripherals.

1.5. System configuration controller

The main purposes of the system configuration controller (SYSCFG) are the following:

- Management of the I/O compensation cell
- Managing the external interrupt line connection to the GPIOs

1.6. System configuration registers

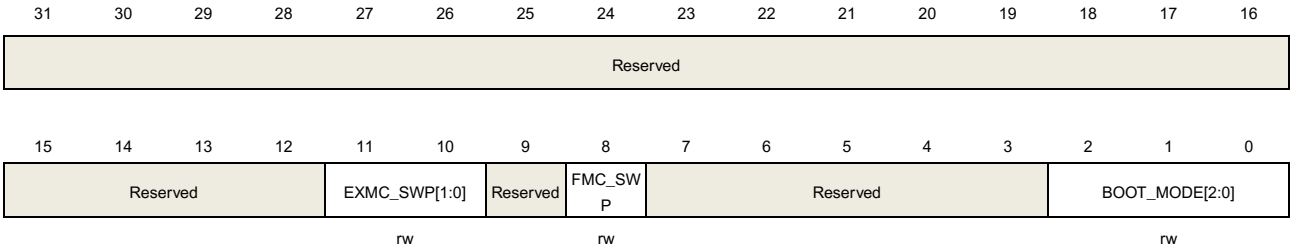
SYSCFG base address: 0x4001 3800

1.6.1. Configuration register 0 (SYSCFG_CFG0)

Address offset: 0x00

Reset value: 0x0000 000X (X indicates BOOT_MODE[2:0] may be any value according to the BOOT0 and BOOT1 pins)

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



Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:10	EXMC_SWP[1:0]	EXMC memory mapping swap These bits control the address mapping swap among the memories in EXMC. 00: No memory mapping swap 01: SDRAM Bank 0 and Bank 1 are swapped with NAND Bank 1 and PC CARD. Then, SDRAM Bank 0 and Bank 1 are mapped at the address region from 0x8000 0000 to 0x9FFF FFFF, NAND Bank 1 is mapped at the address from 0xC000 0000 to 0xCFFF FFFF, and PC CARD is mapped at the address from 0xD000 0000 to 0xDFFF FFFF. Other configurations are reserved.
9	Reserved	Must be kept at reset value.
8	FMC_SWP	FMC memory mapping swap This bit controls the address mapping swap between Bank 0 and Bank 1 of the Main Flash. 0: Main Flash Bank 0 is mapped at address 0x0800 0000 and Main Flash Bank 1 is mapped at address 0x0810 0000(2M FLASH mapping address is 0810_0000, and the 4M FLASH mapping address is 0820_0000) 1: Main Flash Bank 1 is mapped at address 0x0800 0000 and Main Flash Bank 0 is mapped at address 0x0810 0000(2M FLASH mapping address is 0810_0000, and the 4M FLASH mapping address is 0820_0000) **Note:**Setting FMC_SWP in SYSCFG swaps the BANK0 and BANK1 logical addresses in the bus matrix without affecting the original erase address. For the series without BANK1, no swapping.

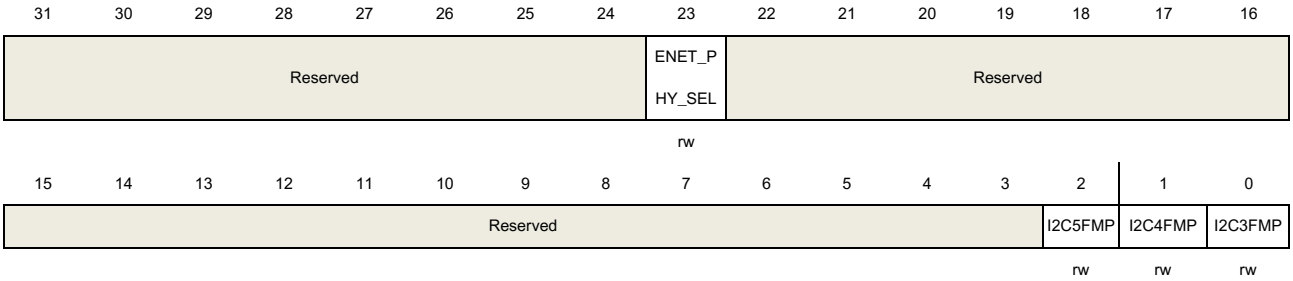
7:3	Reserved	Must be kept at reset value
2:0	BOOT_MODE[2:0]	These bits select the device accessible at address 0x0000 0000. After reset, they take the initial value from the BOOT0 and BOOT1 pins according to the table. *Table 1-3. Boot modes* More devices can be selected by software configuration. Once these bits are written by software, the BOOT0 and BOOT1 pins are ignored. 000: Main Flash memory (0x0800 0000~0x 0877 FFFF) is mapped at address 0x0000 0000 001: Boot loader (0x1FFF 0000~0x1FFF 77FF) is mapped at address 0x0000 0000 011: SRAM0 of on-chip SRAM (0x2000 0000~0x2003 FFFF) is mapped at address 0x0000 000. 101: OTP (0x1FF0 0000 - 0x1FF1 FFFF) is mapped at address 0x0000 0000 Other configurations are reserved. **Note**: Even when mapped at address 0x0000 0000, the related memory is still accessible at its original memory space.

1.6.2. Configuration register 1 (SYSCFG_CFG1)

Address offset: 0x04

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	ENET_PHY_SEL	Ethernet PHY selection This bit selects the PHY interface for the Ethernet MAC. This bit must be configured while the Ethernet MAC is under reset and before the MAC clocks are enabled. 0: MII is selected 1: RMII is selected
22:3	Reserved	Must be kept at reset value.
2	I2C5FMP	Enable Fast mode+ on I2C5
1	I2C4FMP	Enable Fast mode+ on I2C4

0	I2C3FMP	Enable Fast mode+ on I2C3

1.6.3. EXTI sources selection register 0 (SYSCFG_EXTISS0)

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
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 0000: PA3 pin 0001: PB3 pin 0010: PC3 pin 0011: PD3 pin 0100: PE3 pin 0101: PF3 pin 0110: PG3 pin 0111: PH3 pin 1000: PI3 pin Other configurations are reserved.
11:8	EXTI2_SS[3:0]	EXTI 2 sources selection 0000: PA2 pin 0001: PB2 pin 0010: PC2 pin 0011: PD2 pin 0100: PE2 pin 0101: PF2 pin 0110: PG2 pin 0111: PH2 pin 1000: PI2 pin Other configurations are reserved.
7:4	EXTI1_SS[3:0]	EXTI 1 sources selection 0000: PA1 pin 0001: PB1 pin 0010: PC1 pin

		0011: PD1 pin 0100: PE1 pin 0101: PF1 pin 0110: PG1 pin 0111: PH1 pin 1000: PI1 pin Other configurations are reserved.
3:0	EXTI0_SS[3:0]	EXTI 0 sources selection 0000: PA0 pin 0001: PB0 pin 0010: PC0 pin 0011: PD0 pin 0100: PE0 pin 0101: PF0 pin 0110: PG0 pin 0111: PH0 pin 1000: PI0 pin Other configurations are reserved.

1.6.4. EXTI sources selection register 1 (SYSCFG_EXTISS1)

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
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 0000: PA7 pin 0001: PB7 pin 0010: PC7 pin 0011: PD7 pin 0100: PE7 pin 0101: PF7 pin 0110: PG7 pin 0111: PH7 pin

		1000: PI7 pin Other configurations are reserved.
11:8	EXTI6_SS[3:0]	EXTI 6 sources selection 0000: PA6 pin 0001: PB6 pin 0010: PC6 pin 0011: PD6 pin 0100: PE6 pin 0101: PF6 pin 0110: PG6 pin 0111: PH6 pin 1000: PI6 pin Other configurations are reserved.
7:4	EXTI5_SS[3:0]	EXTI 5 sources selection 0000: PA5 pin 0001: PB5 pin 0010: PC5 pin 0011: PD5 pin 0100: PE5 pin 0101: PF5 pin 0110: PG5 pin 0111: PH5 pin 1000: PI5 pin Other configurations are reserved.
3:0	EXTI4_SS[3:0]	EXTI 4 sources selection 0000: PA4 pin 0001: PB4 pin 0010: PC4 pin 0011: PD4 pin 0100: PE4 pin 0101: PF4 pin 0110: PG4 pin 0111: PH4 pin 1000: PI4 pin Other configurations are reserved.

1.6.5. EXTI sources selection register 2 (SYSCFG_EXTISS2)

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
EXTI11_SS[3:0]				EXTI10_SS[3:0]				EXTI9_SS[3:0]				EXTI8_SS[3:0]			
rw				rw				rw				rw			

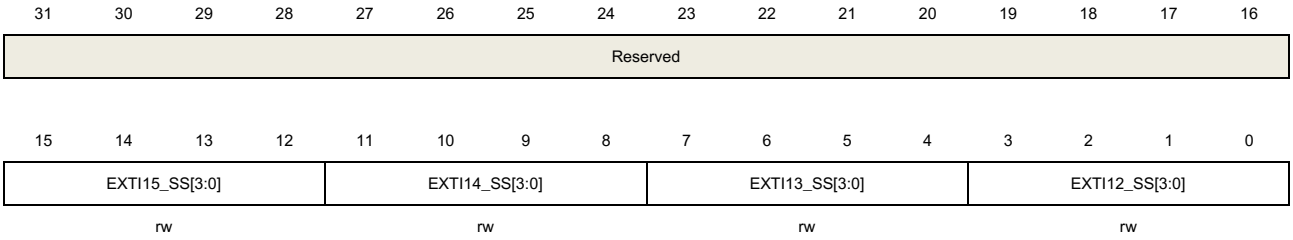
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:12	EXTI11_SS[3:0]	EXTI 11 sources selection 0000: PA11 pin 0001: PB11 pin 0010: PC11 pin 0011: PD11 pin 0100: PE11 pin 0101: PF11 pin 0110: PG11 pin 0111: PH11 pin 1000: PI11 pin Other configurations are reserved.
11:8	EXTI10_SS[3:0]	EXTI 10 sources selection 0000: PA10 pin 0001: PB10 pin 0010: PC10 pin 0011: PD10 pin 0100: PE10 pin 0101: PF10 pin 0110: PG10 pin 0111: PH10 pin 1000: PI10 pin Other configurations are reserved.
7:4	EXTI9_SS[3:0]	EXTI 9 sources selection 0000: PA9 pin 0001: PB9 pin 0010: PC9 pin 0011: PD9 pin 0100: PE9 pin 0101: PF9 pin 0110: PG9 pin 0111: PH9 pin 1000: PI9 pin Other configurations are reserved.
3:0	EXTI8_SS[3:0]	EXTI 8 sources selection

0000: PA8 pin
0001: PB8 pin
0010: PC8 pin
0011: PD8 pin
0100: PE8 pin
0101: PF8 pin
0110: PG8 pin
0111: PH8 pin
1000: PI8 pin
Other configurations are reserved.

1.6.6. EXTI sources selection register 3 (SYSCFG_EXTISS3)

Address offset: 0x14
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	EXTI15_SS[3:0]	EXTI 15 sources selection 0000: PA15 pin 0001: PB15 pin 0010: PC15 pin 0011: PD15 pin 0100: PE15 pin 0101: PF15 pin 0110: PG15 pin 0111: PH15 pin Other configurations are reserved.
11:8	EXTI14_SS[3:0]	EXTI 14 sources selection 0000: PA14 pin 0001: PB14 pin 0010: PC14 pin 0011: PD14 pin 0100: PE14 pin 0101: PF14 pin

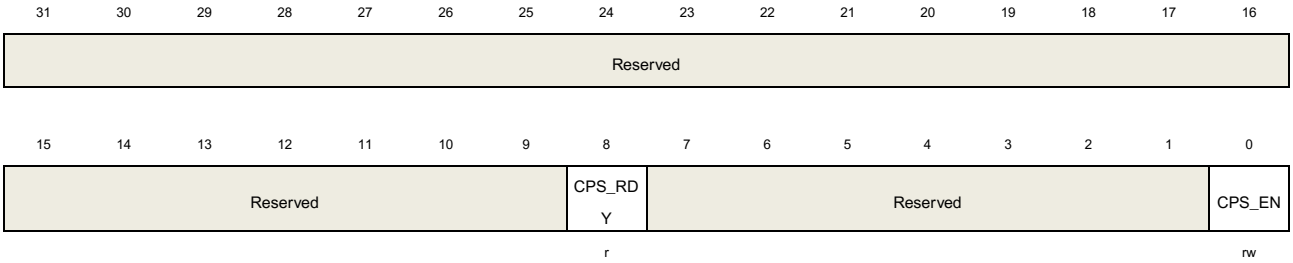
		0110: PG14 pin 0111: PH14 pin Other configurations are reserved.
7:4	EXTI13_SS[3:0]	EXTI 13 sources selection 0000: PA13 pin 0001: PB13 pin 0010: PC13 pin 0011: PD13 pin 0100: PE13 pin 0101: PF13 pin 0110: PG13 pin 0111: PH13 pin Other configurations are reserved.
3:0	EXTI12_SS[3:0]	EXTI 12 sources selection 0000: PA12 pin 0001: PB12 pin 0010: PC12 pin 0011: PD12 pin 0100: PE12 pin 0101: PF12 pin 0110: PG12 pin 0111: PH12 pin Other configurations are reserved.

1.6.7. Compensation cell control register (SYSCFG_CPSCTL)

Address offset: 0x20

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	CPS_RDY	Compensation cell ready flag This bit is read-only. 0: Compensation cell is not ready

		1: Compensation cell is ready
7:1	Reserved	Must be kept at reset value.
0	CPS_EN	Compensation cell power-down 0: I / O compensation cell is power-down 1: I / O compensation cell is enabled

1.6.8. System status register (SYSCFG_STAT)

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	CKMNMIIF	ECCSEIF6	ECCMEIF6	ECCSEIF5	ECCMEIF5	ECCSEIF4	ECCMEIF4	ECCSEIF3	ECCMEIF3	ECCSEIF2	ECCMEIF2	ECCSEIF1	ECCMEIF1	ECCSEIF0	ECCMEIF0
	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
31:15	Reserved	Must be kept at reset value
14	CKMNMIIF	HXTAL clock moniotor NMI interrupt flag The software can clear it by writing 1. 0: no HXTAL clock moniotor error 1: HXTAL clock moniotor is detected.
13	ECCSEIF6	Flash single bit correction event flag The software can clear it by writing 1. 0: no flash single bit correction event is detected. 1:flash single bit correction event is detected.
12	ECCMEIF6	Flash two bits non-correction event flag The software can clear it by writing 1. 0: no flash non-correction event is detected. 1:flash non-correction event is detected.
11	ECCSEIF5	BKPSRAM single bit correction event flag The software can clear it by writing 1. 0: no BKPSRAM single bit correction event is detected. 1: BKPSRAM single bit correction event is detected.
10	ECCMEIF5	Indicates the two bit error BKPSRAM two bits non-correction event flag The software can clear it by writing 1.

		0: no SRAM non-correction event is detected. 1: SRAM non-correction event is detected.
9	ECCSEIF4	TCMSRAM single bit correction event flag The software can clear it by writing 1. 0: no TCMSRAM single bit correction event is detected. 1: TCMSRAM single bit correction event is detected.
8	ECCMEIF4	TCMSRAM two bits non-correction event flag The software can clear it by writing 1. 0: no TCMSRAM non-correction event is detected. 1: TCMSRAM non-correction event is detected.
7	ECCSEIF3	ADDSRAM single bit correction event flag The software can clear it by writing 1. 0: no ADDSRAM single bit correction event is detected. 1: ADDSRAM single bit correction event is detected.
6	ECCMEIF3	ADDSRAM two bits non-correction event flag The software can clear it by writing 1. 0: no ADDSRAM non-correction event is detected. 1: ADDSRAM non-correction event is detected.
5	ECCSEIF2	SRAM2 single bit correction event flag The software can clear it by writing 1. 0: no SRAM2 single bit correction event is detected. 1: SRAM2 single bit correction event is detected.
4	ECCMEIF2	SRAM2 two bits non-correction event flag The software can clear it by writing 1. 0: no SRAM2 non-correction event is detected. 1: SRAM2 non-correction event is detected.
3	ECCSEIF1	SRAM1 single bit correction event flag The software can clear it by writing 1. 0: no SRAM1 single bit correction event is detected. 1: SRAM1 single bit correction event is detected.
2	ECCMEIF1	SRAM1 two bits non-correction event flag The software can clear it by writing 1. 0: no SRAM1 non-correction event is detected. 1: SRAM1 non-correction event is detected.
1	ECCSEIF0	SRAM0 single bit correction event flag The software can clear it by writing 0. 0: no SRAM0 single bit correction event is detected. 1: SRAM0 single bit correction event is detected.
0	ECCMEIF0	SRAM0 two bits non-correction event flag

The software can clear it by writing 1.

0: no SRAM0 non-correction event is detected.

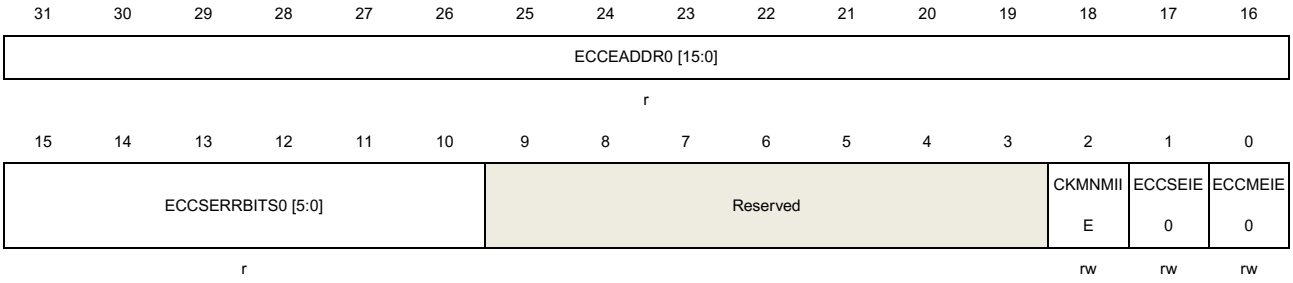
1: SRAM0 non-correction event is detected.

1.6.9. SRAM0 ECC status register (SYSCFG_SRAM0ECC)

Address offset: 0x28

Reset value: 0x0000 0007

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



Bits	Fields	Descriptions
31:16	ECCEADDR0[15:0]	Indicates the last address of ECC event on SRAM0 occurred. **Note:** SRAM0 ECC error word address = SRAM0 base address/4 + ECCEADDR0
15:10	ECCSERRBITS0[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
9:3	Reserved	Must be kept at reset value
2	CKMNMIIE	HXTAL clock moniotor NMI interrupt enable 0: disable 1: enable
1	ECCSEIE0	SRAM0 single bit correction interrupt enable 0: SRAM0 single bit correction interrupt is disabled. 1: SRAM0 single bit correction interrupt is enabled.
0	ECCMEIE0	SRAM0 two bits non-correction interrupt enable 0: SRAM0 non-correction interrupt is disabled. 1: SRAM0 non-correction interrupt is enabled.

1.6.10. SRAM1 ECC status register (SYSCFG_SRAM1ECC)

Address offset: 0x2C

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
ECCEADDR1[13:0]														ECCSERRBITS1	
r														r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECCSERRBITS1[5:0]				Reserved										ECCSEIE	ECCMEIE
														1	1
r														rw	rw

Bits	Fields	Descriptions
31:18	ECCEADDR1[13:0]	Indicates the last address of ECC event on SRAM1 occurred. **Note:** SRAM1 ECC error word address = SRAM1 base address/4 + ECCEADDR1
17:12	ECCSERRBITS1[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
11:2	Reserved	Must be kept at reset value
1	ECCSEIE1	SRAM1 single bit correction interrupt enable 0: SRAM1 single bit correction interrupt is disabled. 1: SRAM1 single bit correction interrupt is enabled.
0	ECCMEIE1	SRAM1 two bits non-correction interrupt enable 0: SRAM1 non-correction interrupt is disabled. 1: SRAM1 non-correction interrupt is enabled.

1.6.11. SRAM2 ECC status register (SYSCFG_SRAM2ECC)

Address offset: 0x30

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
ECCEADDR2[15:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECCSERRBITS2[5:0]				Reserved										ECCSEIE	ECCMEIE
														2	2
r														rw	rw

Bits	Fields	Descriptions
31:16	ECCEADDR2[15:0]	Indicates the address of ECC event on SRAM2 occurred.

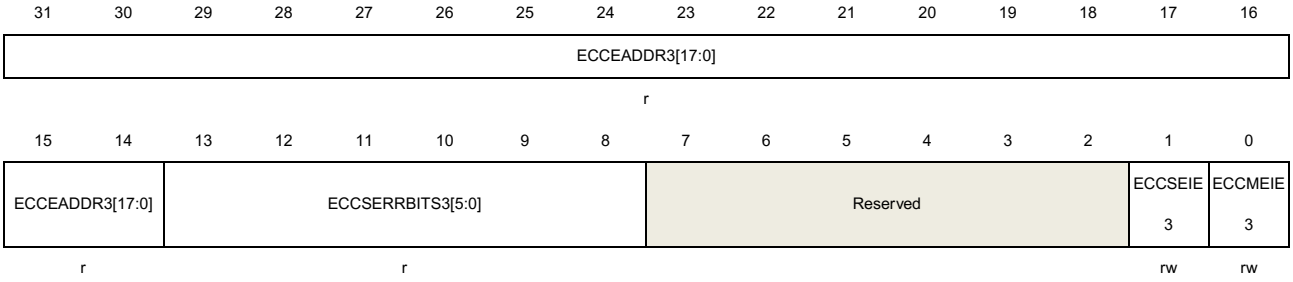
		Note: SRAM2 ECC error word address = SRAM2 base address/4 + ECCEADDR2
15:10	ECCSERRBITS2[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
9:2	Reserved	Must be kept at reset value
1	ECCSEIE2	SRAM2 single bit correction interrupt enable 0: SRAM2 single bit correction interrupt is disabled. 1: SRAM2 single bit correction interrupt is enabled.
0	ECCMEIE2	SRAM2 two bits non-correction interrupt enable 0: SRAM2 non-correction interrupt is disabled. 1: SRAM2 non-correction interrupt is enabled.

1.6.12. ADDSRAM ECC status register (SYSCFG_ADDSRAMECC)

Address offset: 0x34

Reset value: 0x0000 0003

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



Bits	Fields	Descriptions
31:14	ECCEADDR3[17:0]	Indicates the last address of ECC event on ADDSRAM occurred. **Note:** ADDSRAM ECC error word address = ADDSRAM base address/4 + ECCEADDR3
13:8	ECCSERRBITS3[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
7:2	Reserved	Must be kept at reset value

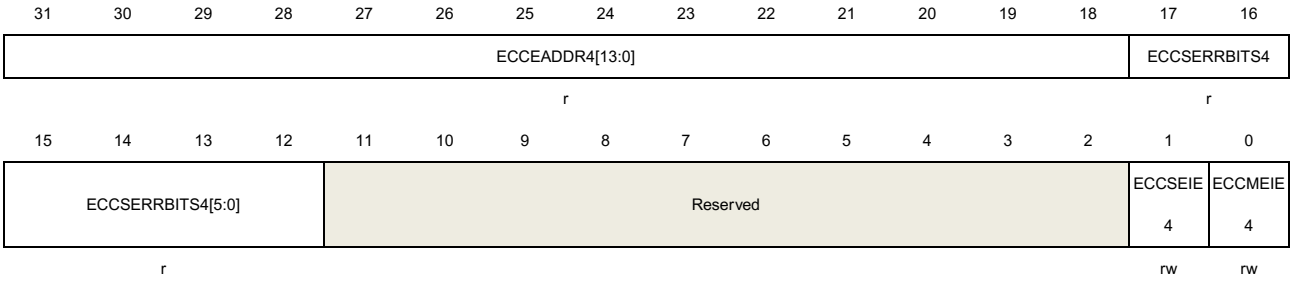
1	ECCSEIE3	ADDSRAM single bit correction interrupt enable 0: ADDSRAM single bit correction interrupt is disabled. 1: ADDSRAM single bit correction interrupt is enabled.
0	ECCMEIE3	ADDSRAM two bits non-correction interrupt enable 0: ADDSRAM non-correction interrupt is disabled. 1: ADDSRAM non-correction interrupt is enabled.

1.6.13. TCMSRAM ECC register (SYSCFG_TCMSRAMECC)

Address offset: 0x38

Reset value: 0x0000 0003

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



Bits	Fields	Descriptions
31:18	ECCEADDR4[13:0]	Indicates the last address of ECC event on TCMSRAM occurred. **Note:** TCMSRAM ECC error word address = TCMSRAM base address/4 + ECCEADDR4
17:12	ECCSERRBITS4[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
11:2	Reserved	Must be kept at reset value
1	ECCSEIE4	TCMSRAM single bit correction interrupt enable 0: TCMSRAM single bit correction interrupt is disabled. 1: TCMSRAM single bit correction interrupt is enabled.
0	ECCMEIE4	TCMSRAM two bits non-correction interrupt enable 0: TCMSRAM non-correction interrupt is disabled. 1: TCMSRAM non-correction interrupt is enabled.

1.6.14. BKPSRAM ECC register (SYSCFG_BKPSRAMECC)

Address offset: 0x3C

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
ECCEADDR5[9:0]										ECCSERRBITS5					
r										r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ECCSEIE5	ECCMEIE5
														rw	rw

Bits	Fields	Descriptions
31:22	ECCEADDR5[9:0]	Indicates the last address of ECC event on BKPSRAM occurred. **Note:** BKPSRAM ECC error word address = BKPSRAM base address/4 + ECCEADDR5
21:16	ECCSERRBITS5[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
15:2	Reserved	Must be kept at reset value
1	ECCSEIE5	BKPSRAM single bit correction interrupt enable 0: BKPSRAM single bit correction interrupt is disabled. 1: BKPSRAM single bit correction interrupt is enabled.
0	ECCMEIE5	BKPSRAM two bits non-correction interrupt enable 0: BKPSRAM non-correction interrupt is disabled. 1: BKPSRAM non-correction interrupt is enabled.

1.6.15. FLASH ECC address register (SYSCFG_FLASHECC_ADDR)

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
ECCEADDR6[31:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECCEADDR6[31:0]															
r															

Bits	Fields	Descriptions
31:0	ECCEADDR6[31:0]	Indicates the last address of ECC event on FLASH occurred. **Note:** FLASH ECC error word address = ECCEADDR6

1.6.16. FLASH ECC register (SYSCFG_FLASHECC)

Address offset: 0x44

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								ECCSERRBITS6[5:0]						ECCSEIE6	ECCMEIE6
								r						rw	rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:2	ECCSERRBITS6[5:0]	Indicates the error bit Which one bit has an ECC single-bit correctable error 0: no error 1: bit 0 … 32: bit 31
1	ECCSEIE6	FLASH single bit correction interrupt enable 0: FLASH single bit correction interrupt is disabled. 1: FLASH single bit correction interrupt is enabled.
0	ECCMEIE6	FLASH two bits non-correction interrupt enable 0: FLASH non-correction interrupt is disabled. 1: FLASH non-correction interrupt is enabled.

1.6.17. User configuration register (USER_CFG)

Address offset: 0x300

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

Bits	Fields	Descriptions
31:24	ANA_VERSION [7:0]	Analog version information
23:0	Reserved	Must be kept at reset value

1.7. Device electronic signature

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

1.7.1. Memory density information

Base address: 0x1FFF 7A20
The value is factory programmed and can never be altered by user.

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

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 density The value indicates the Flash density of the device in Kbytes. Example: 0x0020 indicates 32 Kbytes.
15:0	SRAM_DENSITY [15:0]	SRAM memory density The value indicates the on-chip SRAM memory density of the device in Kbytes. Example: 0x0008 indicates 8 Kbytes.

1.7.2. Unique device ID (96 bits)

Base address: 0x1FFF 7A10
The value is factory programmed and can never be altered by user.

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

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: 0x1FFF 7A14

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNIQUE_ID[63:48]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNIQUE_ID[47:32]															
r															

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

Base address: 0x1FFF 7A18

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UNIQUE_ID[95:80]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNIQUE_ID[79:64]															
r															

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

2. System Security

2.1. Overview

The GD32F527xx is designed with a comprehensive set of system security features. System security features cover several aspects, including firmware intellectual property protection, device private data protection, and service execution assurance.

This chapter systematically introduces the key security features of GD32F527xx and lists the security features available to guide users in building security systems based on GD32F527xx microcontrollers.

2.2. Characteristics

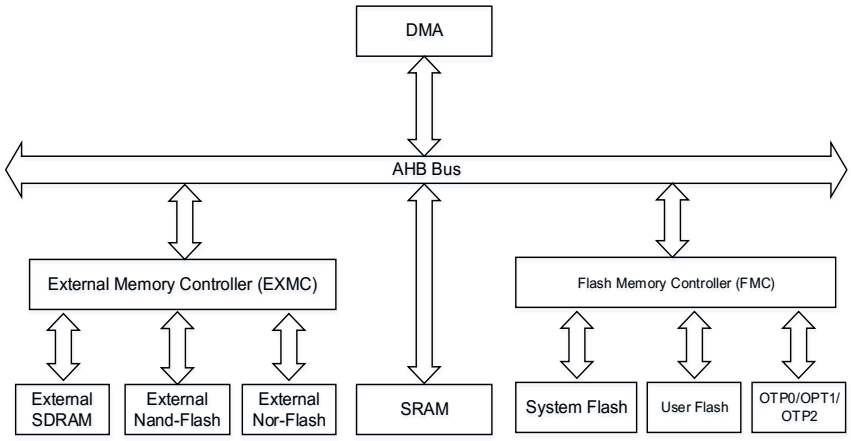
- Memory protections:
  - System FLASH protection.
  - User FLASH protection.
  - SRAM protection.
  - Trusted code protection.
  - Password protection.
  - External SDRAM, Nand-Flash and Nor-Flash protection.
- Boot protection:
  - Unique boot entry.
  - Secure boot.
- Debugging security.
  - Limit debugging.
  - Disable the debugging function.
- Encryption and random numbers.
  - Public key cryptographic acceleration unit (PKCAU).
  - Hash acceleration unit (HAU).
  - Cryptographic acceleration unit (CAU).
- System monitoring:
  - Tamper protection (with RTC).
  - Power supply supervision.
  - Clock security system.
  - Temperature sensor.
  - Critical code execution time monitoring.
- Device UID:
  - Unique 96-bit identifier.

2.3. Memory security

When building system security, memory protection is the most important factor. The memory holds sensitive code and data and should be inaccessible to any unintended interface (debug port) or unauthorized process (insider threat). Depending on the asset (code or data) to be protected, users can set up corresponding protection mechanisms for different storage types (Flash, SRAM, or external memory).

The GD32F527xx microcontroller supports partial access restriction through the memory interface (Flash Memory Controller FMC) or MPU. The purpose of internal memory (Flash and SRAM) and external memory is very different, and the protection mechanism is not the same. The memory access architecture of GD32F527xx is shown in *Figure 2-1. GD32F527xx memory access architecture*.

Figure 2-1. GD32F527xx memory access architecture



2.3.1. System Flash protection

In GD32F527xx, the system memory consists of Bootloader, OPT0, OTP1, and OTP2 regions. The Bootloader area is dedicated to the GD32 MCU Bootloader. To ensure the authenticity and integrity of the Bootloader, users cannot modify this part. Since the bootloader does not contain any sensitive algorithms, it is readable. The OTP (OTP0/1/2) is a one-time programming region, and all OTP regions support write locking to prevent illegal write operations. OTP1/2 also supports read locking to prevent illegal read operations. Therefore, OTP1 can be used as a trusted code region and OTP2 can be used as a password region.

2.3.2. User Flash protection

The user flash is generally used to store user code and important data. Users set memory security properties to protect specific areas through FMC or MPU.

External read access protection

Flash memory controller (FMC) provides a security protection feature to prevent illegal reading of flash memory, which can protect software and firmware from illegal user operations. FMC offer three levels of safety:

No protection: When setting EFSPC = 0 in EFUSE control and SPC byte to 0xAA, no protection performed. User flash and option byte block are accessible by all operations.

Protection level low: When setting EFSPC = 1 in EFUSE control or SPC byte to any value except 0xAA or 0xCC, protection level low performed. The user flash can only be accessed by user code. In debug mode, boot from SRAM or boot from boot loader mode, all operations to user flash are forbidden. If a read operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, a bus error will be generated. At protection level low, option byte block is accessible by all operations. If program back to no protection level by setting SPC byte to 0xAA, a mass erase for user flash will be performed.

Note: When EFSPC = 1 in the EFUSE control segment takes effect, the system cannot be rolled back.

Protection level high: when setting SPC byte 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 user flash block is accessible by all operations from user code. The SPC byte cannot be reprogrammed.

For details about the security protection of the user Flash, refer to ***Security protection***.

Note: If protection level high is programmed, it cannot move back to protection level low or no protection level, and option bytes cannot be programmed again.

Internal erase / programming protection

Read or write access inside the user flash may be initiated by malware injected into the device SRAM. Therefore, users can set the protection properties of Flash to prevent from tampering with sensitive data or code via the unexpected write operations.

The erase / programming protection of the user flash can be achieved in two ways:

1. Protecting the target sector by configuring option bytes through the FMC. The sector erase / programming protection of FMC prevents accidental manipulation of flash memory. If the erase / programming protection interrupt is turned on, erasing or programming the protected area triggers an error interrupt. Users can execute the corresponding protection policy in the interrupt.
2. Configure the access properties of the user flash using the MPU. Set the access attribute of the target area to deny write access. When writing to the protected area, the MCU will generate an exception, and user can execute the corresponding protection policy in the exception.

Note: Write protection should also be set for unused flash areas to prevent code modification

or injection.

Error Checking and Correction (ECC)

GD32F527xx series support hardware ECC function for flash, user flash can use ECC to achieve error detection and correction (two-bit error detection, one error correction). As a functional safety mechanism, ECC can also be used for safety protection as a complementary mechanism to prevent fault injection.

2.3.3. SRAM protection

Code execution

GD32F527xx series user flash supports a maximum of 2M space of zero waiting area, which has excellent execution efficiency. Generally, when the code does not exceed 2M size, it is not necessary to copy the code in flash to SRAM for execution. However, because SRAM doesn't have to wait, when the user's code size exceeds 2M. if a piece of code needs faster performance, it can be copied from the user flash into SRAM for execution.

Therefore, it is necessary to protect the area of code executed in SRAM and access properties through MPU configuration. If no code needs to be executed in SRAM, it is recommended that the MPU accurately configure its properties to never execute to prevent malicious programs from running.

Data clearing

The SRAM may store sensitive data or temporary values from which confidential information can be obtained. As a typical example, keys located in protected areas are temporarily stored in the SRAM in clear text when they need to be transferred. It is highly recommended to clear the working cache and variables immediately after the function has finished manipulating sensitive data.

2.3.4. Trusted code protection

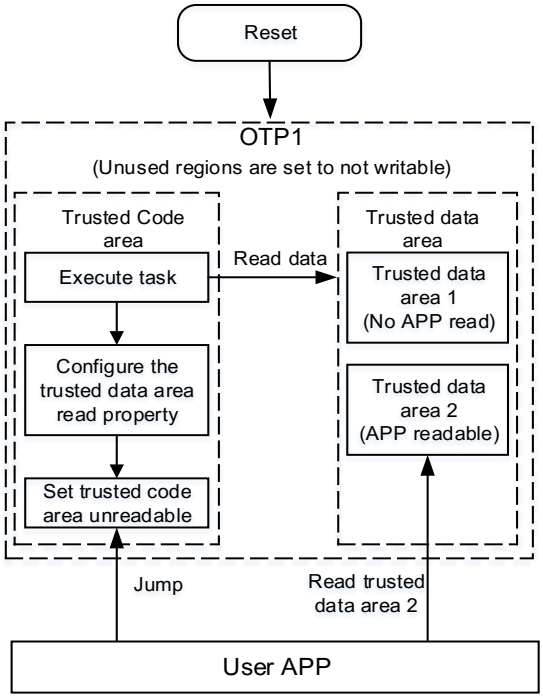
The OTP1 area of GD32F527xx can be used as the user's trusted code area. The OTP1 area is divided into 16 data blocks of 8K bytes and one lock block of 16 bytes. This area supports lock operations and prevents illegal read / write operations. Each lock byte in a lock block can lock a corresponding block of data to prevent programming operations on those blocks. The OTP1REN[15:0] bit in FMC_OPT1CFG register determines whether the OTP1 data block is readable. Reading a locked region causes a bus error.

The user's bootloader and important data can be stored in this area, and the MCU can be set to boot from the trusted code area. After the trusted code area completes the task, user can lock the code area as forbidden to read before jumping to the APP. Data area attributes can be flexibly configured based on user application scenarios. The data can be locked after the execution of the trusted code, and the data read and write properties can be set by region.

For example, some data can be read by the APP, but some data cannot be read by the APP.

The schematic diagram of the protection of the trusted code area is shown in ***Figure 2-2. Trusted code area protection***, and the details of OTP1 can be referred to ***OTP block programming***.

Figure 2-2. Trusted code area protection



Note: Since the OPT1 region can only be written once, the user cannot modify the data block again when it is locked as no write. The state of the read property can be switched through FMC registers.

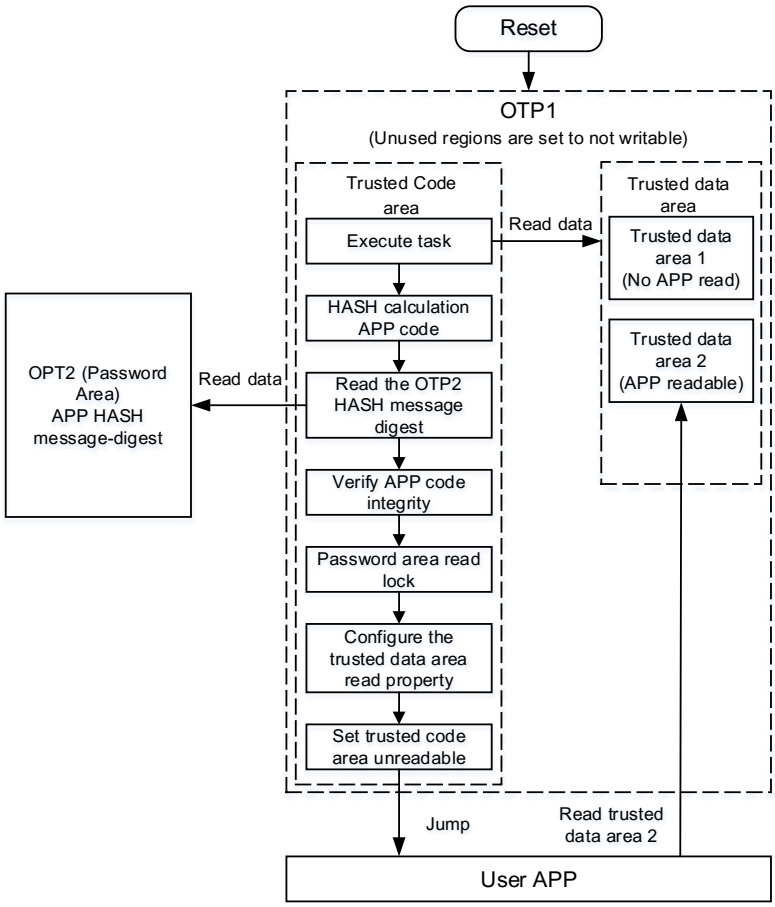
2.3.5. Password protection

The OTP2 area of GD32F527xx can be used as the user's key area. OTP2 block can be divided to 16 data blocks which has 32 bytes each and 1 lock block which has 32 bytes. OTP2 supports write locking and read locking. The user can configure read locking for the target region first, and then by setting the RLBE bit of the FMC_CTL register to 1, the target region will be unable to perform read access.

Users can put a key or HASH message digest into the OTP2 area (password area), which can only be read by trusted code, not APP code. A typical application is the OTP2 area where HASH message-digest of APP code are stored. After the user starts from the trusted code in OTP1 area, the user uses the hardware HASH processor to calculate the APP firmware, and then verifies the hash message digest in OTP2 area to ensure the integrity of the APP firmware. The trusted code can configure RLBE bit to lock read of OTP2 area, and then jump to the APP firmware. Data blocks corresponding to the OTP2 read lock block cannot be read by the APP code. It cannot be read by trusted code until the next time the MCU resets.

For details on OTP2 refer to ***OTP block programming***.

Figure 2-3. Trusted code area protection



2.3.6. External memory protection

If the external memory is connected to the microcontroller via a dedicated EXMC interface, the external memory stores code and data just like the internal storage, but the external memory introduces issues such as confidentiality and authentication.

External memory can be mapped to the address of External RAM and External Device (Refer to ***Table 1-2. Memory map of GD32F527xx devices***). User can configure the access attribute of this area through the MPU to protect the data of external memory. When external memory is set to write protection, the data is protected from being erased or modified. If the data of external memory need to be confidential, TRNG, PKCAU and CAU hardware can be encrypted.

2.4. Boot protection

Boot protection protects the first software instruction in the system. If an attacker successfully tampers with the MCU boot address, they can execute their own code to bypass the MCU

initial protection configuration, or execute insecure programs to access the MCU memory.

The GD32F527xx series can be configured for boot, with the option to perform boot in the firmware in the user flash, system bootloader, or SRAM. Boot protection depends on the uniqueness of the MCU boot entry, which executes trusted code.

2.4.1. Unique boot entry

The GD32F527xx series can configure flash or OTP1 (Trusted code area) as the only boot entry and cannot be modified again. The two boot portals can be configured as follows:

1. If the BTFOSEL value of EFUSE is 0, the FMC sets the SPC protection level to high and can only be started from Flash, not from SRAM or Bootloader.
2. If the BTFOSEL of EFUSE is 1, the FMC sets the SPC protection level to high and can only be started from OTP1, not from SRAM, Flash, or Bootloader.

Since the BTFOSEL bit of EFUSE cannot be rolled back after it is written to 1 and takes effect, the same is true when SPC is set to a high protection level. These two security attributes ensure the uniqueness of the MCU boot entry.

2.4.2. Secure boot

The secure boot executes before the user APP firmware when the MCU resets, and it provides initial secure features. Users can store user-level secure boot in OTP1 area, and then set OTP1 area as the unique entry of MCU. After each reset, the MCU must be booted from the secure boot.

The main functions of secure boot include:

1. Check the MCU security configuration and set up runtime protection.

   Secure boot checks whether the static security configuration is correct by configuring option bytes (security protection level SPC, read protection DRP, erase / programming protection WP).

   Runtime protection can be achieved through the configuration of MPU, Temper Detection and FWDGT.

2. Verify the integrity and authenticity of the APP firmware.

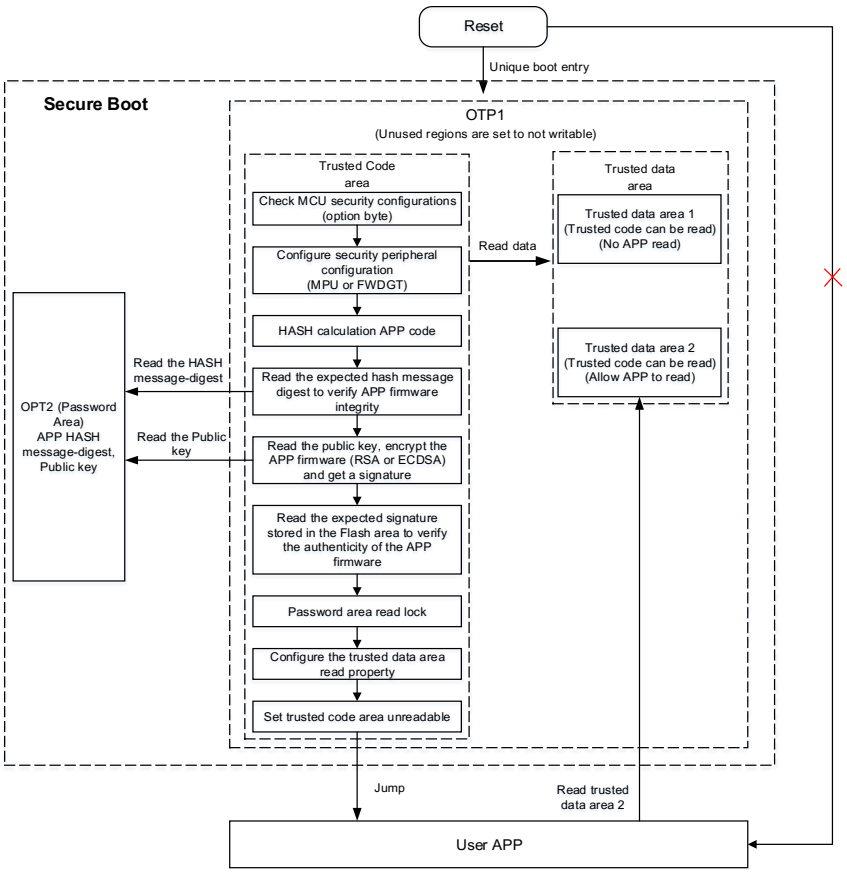
   Users can calculate the APP code using the HASH algorithm (MD5, SHA1, SHA-224, or SHA256 hash algorithms) and compare the calculated summary with the expected value to confirm the integrity of the APP code.

   The signature obtained by encrypting the APP firmware through the shared key between the user and the MCU (PKCAU supports RSA and ECDSA algorithms), and is compared with the expected signature to realize authenticity check.

HASH message-digest, shared keys, and APP firmware signatures should be stored in protected area. HASH message-digest and shared key can be stored in OPT2 area, and APP firmware signature can be stored in user flash area.

The process for a secure startup is shown in ***Figure 2-4. Secure boot process***.

Figure 2-4. Secure boot process



2.5. Debugging security

Debug ports provide access to internal resources (kernel, memory, and peripheral register) and should be disabled in the final product. Debug port access to MCU data is the most basic external attack, and users can avoid it simply by disabling the JTAG (or SWD) port or permanently disabling the debug function with secure and immutable firmware.

2.5.1. Limit debugging

Users can set the flash security protection level with the option byte (SPC) or EFUSE.

1. When EFSPC in the EFUSE control segment is set to 1 or the SPC byte is set to any value other than 0xAA or 0xCC, the low protection level is activated.
2. If the SPC byte is set to 0xCC, the high protection level is activated.

When the MCU is in the above two security levels, the main flash block can only be accessed by user code, and cannot be accessed by user code through debug mode. However, MCU registers can still be accessed through the debugger.

2.5.2. Disable debugging function

The GD32F527xx series supports permanently disabling the debugging function. Users can permanently disable the debugging function by setting the NDBG bit of EFUSE to 1. When EFUSE takes effect, the NDBG bit cannot be rolled back. Therefore, turning off debugging is recommended for use in the final stages of production.

2.6. Encryption and random numbers

Encryption and decryption algorithms are very important to ensure the security of embedded systems. Encryption and decryption algorithms can ensure the confidentiality, integrity and authenticity of data or code. To efficiently support encryption and decryption, the GD32F527xx family provides Public Key Cryptographic Acceleration Unit (PKCAU), Cryptographic Acceleration Unit (CAU), Hash Acceleration Unit (HAU), and True Random number generator (TRNG) peripherals.

Public Key Cryptographic Acceleration Unit (PKCAU)

- Support RSA / DH algorithms with up to 3136 bits of operands.
- Support ECC algorithm with up to 640 bits of operands.
- RSA modular exponentiation, RSA CRT exponentiation.
- ECC scalar multiplication, check point on elliptic curve.
- ECDSA (Elliptic Curve Digital Signature Algorithm) signature and verification.
- Support Montgomery multiplication, accelerate RSA, DH and ECC operations.

For details on PKCAU peripherals refer to ***Public Key Cryptographic Acceleration Unit (PKCAU)***.

Cryptographic Acceleration Unit (CAU)

- DES, TDES and AES encryption / decryption algorithms are supported.

DES / TDES

- Supports the ECB and CBC chaining algorithms.
- two 32-bit initialization vectors (IV) are used in CBC mode.
- 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.
- Data can be transferred by DMA, CPU during interrupts, or without both of them.

For details on CAU peripherals refer to ***Cryptographic Acceleration Unit (CAU)***.

Hash Acceleration Unit (HAU)

- Federal Information Processing Standards Publication 180-2 (FIPS PUB 180-2).
- Secure Hash Standard specifications (SHA-1, SHA-224, SHA-256).
- Internet Engineering Task Force Request for Comments number 1321 (IETF RFC 1321) specifications (MD5).
- DMA transfer is supported.

For details on HAU peripherals refer to ***Hash Acceleration Unit (HAU)***.

True random number generator (TRNG)

- 32-bit random value seed is generated from analog noise, so the random number is a true random number.

For details on TRNG peripherals refer to ***True random number generator (TRNG)***.

2.7. System monitoring

Users can avoid faults by monitoring the MCU's power supply, clock and other environments. When the environment changes, corresponding countermeasures can be taken to ensure the security of the system.

2.7.1. Tamper protection (with RTC)

Tamper detection is used to detect system-level or board-level intrusions. The open cap is detected on the MCU pin and triggers the appropriate action. The TAMP detection of the RTC is used to detect physical tampering on the system. When the tamper detection occurs, the information saved in the RTC_BKPx register can be erased.

2.7.2. Power supply supervision

Power voltage monitors are used to detect abnormal levels. Below a certain voltage, normal operation is not guaranteed, and this may be a sign of a fault injection attack.

Some attacks may target the microcontroller power supply to cause errors that can render security countermeasures ineffective. GD32F527xx supports low voltage detection (LVD), triggering low voltage interruption, and users can make corresponding countermeasures in the low voltage interruption.

2.7.3. Clock security system

Clock safety systems are used to prevent the failure of external oscillators. If a fault is detected on the external clock, the microcontroller switches to use the internal clock to safely perform the operation.

Missing clock sources can be intentional or unintentional. In either case, the device must take

appropriate measures to recover. The GD32F527xx supports external clock fault detection. When the external clock is lost, users can switch to the internal clock source in time through the NMI exception.

2.7.4. Temperature sensor

The temperature increase may be a way of fault injection attack scheme. GD32F527xx built-in temperature sensor, can obtain MCU temperature changes through the internal ADC channel, the user can set the temperature range according to the application. When the temperature is detected beyond the preset range, the corresponding countermeasures are implemented.

2.7.5. Critical code execution time monitoring

FWDGT is generally used to solve code faults or jams. When FWDGT reaches a specified timeout, the system will reset. FWDGT has a separate internal low-speed clock, which can be used to control the execution time of critical code, such as encryption or flash programming.

2.8. Device UID

Each GD32 MCU has a unique 96-bit identifier that provides a single reference to any device in any environment. The user can never modify, and the MCU's unique identifier can be used to directly verify device identity.

3. Flash memory controller (FMC)

3.1. Overview

The flash memory controller, FMC, provides all the necessary functions for the on-chip flash memory. There is no waiting time while CPU executes instructions stored in the first 2048K bytes of the flash. It provides page (4KB) erase, sector erase, mass erase, and double word/word/half-word/byte program operations for flash memory.

The EFUSE controller has efuse macro that store system paramters. As a non-volatile unit of storage, the bit of efuse macro cannot be restored to 0 once it is programmed to 1. According to the software operation, the EFUSE controller can program all bits in the system parameters.

3.2. Characteristics

- Up to 7680KB of main flash memory for instruction and data.
- No waiting time within first 2048K bytes when CPU executes instructions (in case that flash size less than 2048K, all memory is no waiting time). A long delay when CPU fetches the instructions out of the range.
- 2 banks adopted for GD32F527xx. Bank0 is used for the first 2048KB and bank1 is for the rest capacity.
- ECC with single bit error corrected and double bit errors detected.
- Double word programming, word/half-word/byte programming, page (4KB) erase, sector erase and mass erase operation.
- Two option bytes blocks size of 16B for user application requirements.
- 64B OTP0 (One-time program) block used for user data storage. Additional 128K bytes OTP1 and 128B OTP2.
- 30K bytes information block for bootloader.
- Option bytes are uploaded to the option byte control registers on every system reset.
- Flash security protection to prevent illegal code/data access.
- Page erase/program protection to prevent unexpected operation.
- One-time programmable nonvolatile EFUSE storage cells.
- All bits in the EFUSE cannot be rollback from 1 to 0.
- EFUSE can only be accessed through corresponding register.

3.3. Function overview

3.3.1. Flash memory architecture

For GD32F527xx with flash no more than 7680KB, with 16K bytes of 8 sectors, 64K bytes of 2 sectors, 128K bytes of 30 sectors, 256K bytes of 14 sectors. Each sector can be erased individually.

The flash memory structure is divided into 4MB dual bank, 2MB dual bank, 1MB single bank, and 512KB single bank. Each type of structure can have an extended Bank1 flash memory (Bank1_Ex), with the address of the Bank1_Ex fixed to start at 0x0840 0000 and the operation mode being the same as Bank1. The details of the 4MB dual-bank structure are shown in ***Table 3-1. GD32F527xx base address and size for 4MB dual bank flash memory***. The 2MB dual-bank structure is detailed in ***Table 3-2. GD32F527xx base address and size for 2MB dual bank flash memory***. The 1MB single-bank structure details are shown in ***Table 3-3. GD32F527xx base address and size for 1MB single bank flash memory***. The details of the 512KB single-bank structure are shown in ***Table 3-4. GD32F527xx base address and size for 512KB single bank flash memory***.

Table 3-1. GD32F527xx base address and size for 4MB dual bank flash memory

Block		Name	Address	size(bytes)
Main Flash Block	Bank0 2MB	Sector 0	0x0800 0000 - 0x0800 3FFF	16KB
		Sector 1	0x0800 4000 - 0x0800 7FFF	16KB
		Sector 2	0x0800 8000 - 0x0800 BFFF	16KB
		Sector 3	0x0800 C000 - 0x0800 FFFF	16KB
		Sector 4	0x0801 0000 - 0x0801 FFFF	64KB
		Sector 5	0x0802 0000 - 0x0803 FFFF	128KB
		Sector 6	0x0804 0000 - 0x0805 FFFF	128KB
		. . .	. . .	. . .
		Sector 19	0x081E 0000 - 0x081F FFFF	128KB
	Bank1 2MB	Sector 20	0x0820 0000 - 0x0820 3FFF	16KB
		Sector 21	0x0820 4000 - 0x0820 7FFF	16KB
		Sector 22	0x0820 8000 - 0x0820 BFFF	16KB
		Sector 23	0x0820 C000 - 0x0820 FFFF	16KB
		Sector 24	0x0821 0000 - 0x0821 FFFF	64KB
		Sector 25	0x0822 0000 - 0x0823 FFFF	128KB
		Sector 26	0x0824 0000 - 0x0825 FFFF	128KB
		. . .	. . .	. . .
		Sector 39	0x083E 0000 - 0x083F FFFF	128KB
	Bank1_Ex 3584KB	Sector40	0x0840 0000 - 0x0843 FFFF	256KB
		Sector41	0x0844 0000 - 0x0847 FFFF	256KB
		. . .	. . .	. . .
		Sector53	0x0874 0000 - 0x0877 FFFF	256KB
Information Block		Bootloader	0x1FFF 0000- 0x1FFF 77FF	30KB
OTP0 Block		data area	0x1FFF 7800 - 0x1FFF 783F	64B
		lock area	0x1FFF 7840 - 0x1FFF 787F	64B

Block	Name	Address	size(bytes)
OTP1 Block	data area	0x1FF0 0000 - 0x1FF1 FFFF	128KB
	lock area	0x1FF2 0200 - 0x1FF2 020F	16B
OTP2 Block	data area	0x1FF2 0000 - 0x1FF2 01FF	512B
	lock area	0x1FF2 0210 - 0x1FF2 022F	32B
Option bytes Block	Bank0 option	0x1FFF C000 - 0x1FFF C00F	16B
	Bank1 option	0x1FFE C000 - 0x1FFE C00F	16B

NOTE:

1. The Information Block stores the boot loader. This block cannot be programmed or erased by user.

2. For 4MB dual bank and 2MB dual bank structure, setting FMC_SWP in SYSCFG swaps the BANK0 and BANK1 logical addresses in the bus matrix without affecting the original erase address. For example, if setting FMC_SWP to '1', the context in 0x0800 0000 can be erased by page erase (PE_ADDR=0x0820 0000), sector erase (Sector 20) or mass erase (MER1=1). If setting FMC_SWP to '0', the context in 0x0800 0000 can be erased by page erase (PE_ADDR=0x0800 0000), sector erase (Sector 0) or mass erase (MER0=1).

3. For the 1MB single bank or 512KB single bank structure, the FMC_SWP switching function is not supported.

Table 3-2. GD32F527xx base address and size for 2MB dual bank flash memory

Block		Name	Address	size(bytes)
Main Flash Block	Bank0 1MB	Sector 0	0x0800 0000 - 0x0800 3FFF	16KB
		Sector 1	0x0800 4000 - 0x0800 7FFF	16KB
		Sector 2	0x0800 8000 - 0x0800 BFFF	16KB
		Sector 3	0x0800 C000 - 0x0800 FFFF	16KB
		Sector 4	0x0801 0000 - 0x0801 FFFF	64KB
		Sector 5	0x0802 0000 - 0x0803 FFFF	128KB
		Sector 6	0x0804 0000 - 0x0805 FFFF	128KB
		. . .	. . .	. . .
		Sector 11	0x080E 0000 - 0x080F FFFF	128KB
	Bank1 1MB	Sector 20	0x0810 0000 - 0x0810 3FFF	16KB
		Sector 21	0x0810 4000 - 0x0810 7FFF	16KB
		Sector 22	0x0810 8000 - 0x0810 BFFF	16KB
		Sector 23	0x0810 C000 - 0x0810 FFFF	16KB
		Sector 24	0x0811 0000 - 0x0811 FFFF	64KB
		Sector 25	0x0812 0000 - 0x0813 FFFF	128KB
		Sector 26	0x0814 0000 - 0x0815 FFFF	128KB
		. . .	. . .	. . .

Block		Name	Address	size(bytes)
		Sector 31	0x081E 0000 - 0x081F FFFF	128KB
	Bank1_Ex 3584KB	Sector40	0x0840 0000 - 0x0843 FFFF	256KB
		Sector41	0x0844 0000 - 0x0847 FFFF	256KB
		.	.	.
		.	.	.
		.	.	.
		Sector53	0x0874 0000 - 0x0877 FFFF	256KB

Table 3-3. GD32F527xx base address and size for 1MB single bank flash memory

Block		Name	Address	size(bytes)
Main Flash Block	Bank0 1MB	Sector 0	0x0800 0000 - 0x0800 3FFF	16KB
		Sector 1	0x0800 4000 - 0x0800 7FFF	16KB
		Sector 2	0x0800 8000 - 0x0800 BFFF	16KB
		Sector 3	0x0800 C000 - 0x0800 FFFF	16KB
		Sector 4	0x0801 0000 - 0x0801 FFFF	64KB
		Sector 5	0x0802 0000 - 0x0803 FFFF	128KB
		Sector 6	0x0804 0000 - 0x0805 FFFF	128KB
		.	.	.
		.	.	.
		.	.	.
		Sector 11	0x080E 0000 - 0x080F FFFF	128KB
	Bank1_Ex 3584KB	Sector40	0x0840 0000 - 0x0843 FFFF	256KB
		Sector41	0x0844 0000 - 0x0847 FFFF	256KB
		.	.	.
		.	.	.
		.	.	.
		Sector53	0x0874 0000 - 0x0877 FFFF	256KB

Table 3-4. GD32F527xx base address and size for 512KB single bank flash memory

Block		Name	Address	size(bytes)
Main Flash Block	Bank0 512KB	Sector 0	0x0800 0000 - 0x0800 3FFF	16KB
		Sector 1	0x0800 4000 - 0x0800 7FFF	16KB
		Sector 2	0x0800 8000 - 0x0800 BFFF	16KB
		Sector 3	0x0800 C000 - 0x0800 FFFF	16KB
		Sector 4	0x0801 0000 - 0x0801 FFFF	64KB
		Sector 5	0x0802 0000 - 0x0803 FFFF	128KB
		Sector 6	0x0804 0000 - 0x0805 FFFF	128KB
		Sector 7	0x0806 0000 - 0x0807 FFFF	128KB
	Bank1_Ex 3584KB	Sector40	0x0840 0000 - 0x0843 FFFF	256KB
		Sector41	0x0844 0000 - 0x0847 FFFF	256KB
		.	.	.
		.	.	.
		.	.	.

Block		Name	Address	size(bytes)
		Sector53	0x0874 0000 - 0x0877 FFFF	256KB

3.3.2. Error Checking and Correcting (ECC)

The ECC mechanism supports:

- One error detection and correction
- Two errors detection

ECCEN bit of the option byte determines whether to enable ECC.

When one error is detected and corrected:

- When occurred in reading from main flash / bootloader / OTP0 / OTP1 / OTP2, the ECCSEIF6 bit in SYSCFG_STAT register will be set. If the ECCSEIE6 bit in SYSCFG_FLASH_ECC register is set, an IRQ101 interrupt is generated. The ECCEADDR6[31:0] in SYSCFG_FLASHECC_ADDR register and ECCSERRBITS6[5:0] in SYSCFG_FLASH_ECC register notice error offset address and bits respectively.

When two errors are detected:

- When occurred in load code from main flash / bootloader / OTP1, the LDECCDET bit in FMC_STAT register will be set. If the LDECCIE bit in FMC_CTL register is set, an NMI interrupt is generated. FMC_LDECCADDR0 / FMC_LDECCADDR1 / FMC_LDECCADDR2 registers will notice three error offset addresses in order. Must use the double word programming main flash/ bootloader / OTP1 to ensure that this error is detected correctly.
- When occurred in reading from main flash / bootloader / OTP0 / OTP1 / OTP2, the ECCMEIF6 bit in SYSCFG_STAT register will be set. If the ECCMEIE6 bit in SYSCFG_FLASH_ECC register is set, an NMI interrupt is generated. The ECCEADDR6[31:0] in SYSCFG_FLASHECC_ADDR register and ECCSERRBITS6[5:0] in SYSCFG_FLASH_ECC register notice error offset address and bits respectively.

Note:

1. Data in Flash memory are 72-bits words: 8 bits are added per double word (64 bits), but the added 8bits are calculated by hardware and can not be accessed by user.
2. Erase operation is required before programming (Before the first programming, the original data is all F, and the area must also be erased before programming)..

3.3.3. Read operations

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

3.3.4. Unlock the FMC_CTL/FMC_OBCTLx 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 0x4567 0123 and 0xCDEF 89AB 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_OBCTL0 registers are still protected even the FMC_CTL is unlocked. The unlocking sequence is two write operations, which are writing 0x0819 2A3B and 0x4C5D 6E7F to FMC_OBKEY register. After the two write operations, the OB_LK bit in FMC_OBCTL0 register is reset to 0 by hardware. The software can lock the FMC_OBCTLx again by setting the OB_LK bit in FMC_OBCTLx register to 1.

3.3.5. Page erase

The FMC provides additional page (4KB) erase function which is used to initialize the contents of a main flash memory page to a high state. Each page can be erased independently without affecting the contents of other pages. The following steps show the access sequence of the registers for a page erase operation.

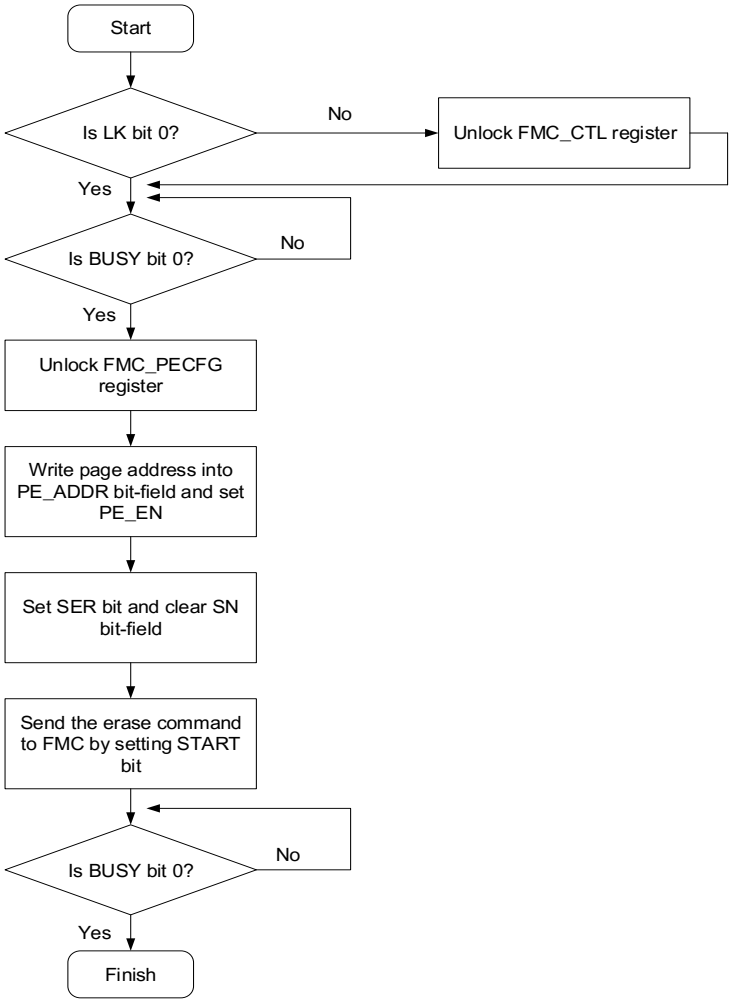
1. Unlock the FMC_CTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
3. Write Key value 0xA9B8 C7D6 in the FMC_PEKEY register to unlock the FMC_PECFG register.
4. Set the PE_EN bit in the FMC_PECFG register to enable page erase function.
5. Write the page address in the PE_ADDR[28:0] bit-field in the FMC_PECFG register. The page address need to follow 4K-byte alignment.
6. Make sure the value of the SN[4:0] bit-field is 0 and set the SER bit in FMC_CTL register.
7. Send the sector erase command to the FMC by setting the START bit in FMC_CTL register.
8. Wait until all the operations have finished by checking the value of the BUSY bit in FMC_STAT register.
9. Clear the PE_EN bit and SER bit to prevent misoperation next time.
10. Read and verify the sector if required using a CBUS access.

Note: Do not power off or reset during the erase process.

When ENDIE bit in the FMC_CTL register is set and the page erase operation is executed successfully, the END in FMC_STAT register is set, and an interrupt will be triggered. Note that a correct target page address (4KB alignment) must be confirmed. Or the software may run out of control if the target erase page is being used to fetch codes or to access data. The FMC will not provide any notification when this occurs. Additionally, the page erase operation

will be ignored on erase / program protected sectors. In this condition, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the OPERR bit in the FMC_STAT register to detect this condition in the interrupt handler. ***Figure 3-1. Process of page erase operation*** shows the page erase operation flow.

Figure 3-1. Process of page erase operation



3.3.6. Sector erase

The FMC provides a sector erase function which is used to initialize the contents of a main flash memory 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.

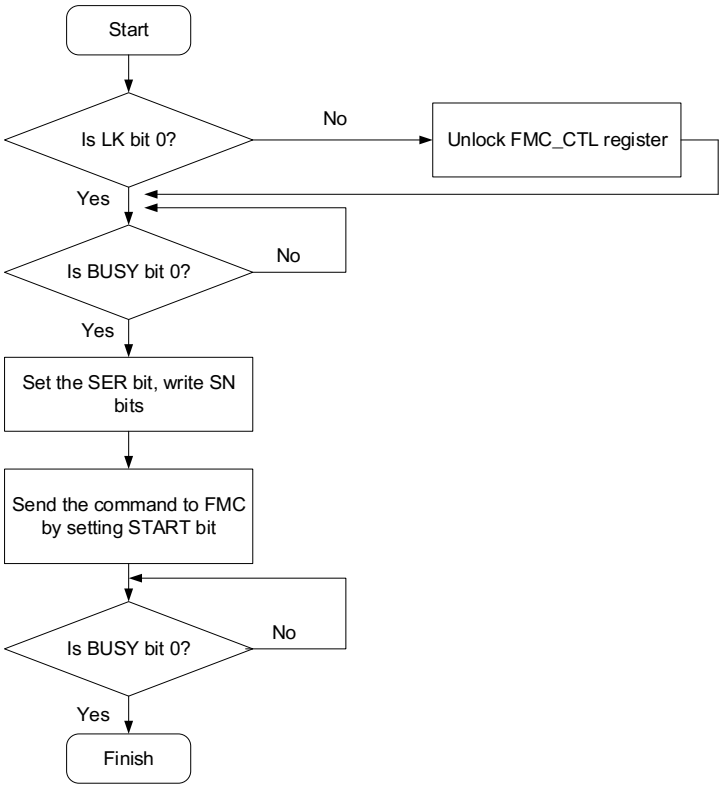
1. Unlock the FMC_CTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
3. Set the SER bit in FMC_CTL register.

4. Write the sector number SN bits in the FMC_CTL 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 using a CBUS access.

Note: Do not power off or reset during the erase process.

When ENDIE bit in the FMC_CTL register is set and the sector erase operation is executed successfully, the END 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 number 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 flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the OPERR bit in the FMC_STAT register to detect this condition in the interrupt handler. *Figure 3-2. Process of sector erase operation* shows the sector erase operation flow.

Figure 3-2. Process of sector erase operation



3.3.7. Mass erase

The FMC provides a complete erase function which is used to initialize the main flash block contents. This erase can affect only on Bank0 by setting MER0 bit to 1, or only on

Bank1(include Bnak1_Ex) by setting MER1 bit to 1, or on entire flash by setting MER0 and MER1 bits to 1. The following steps show the mass erase register access sequence.

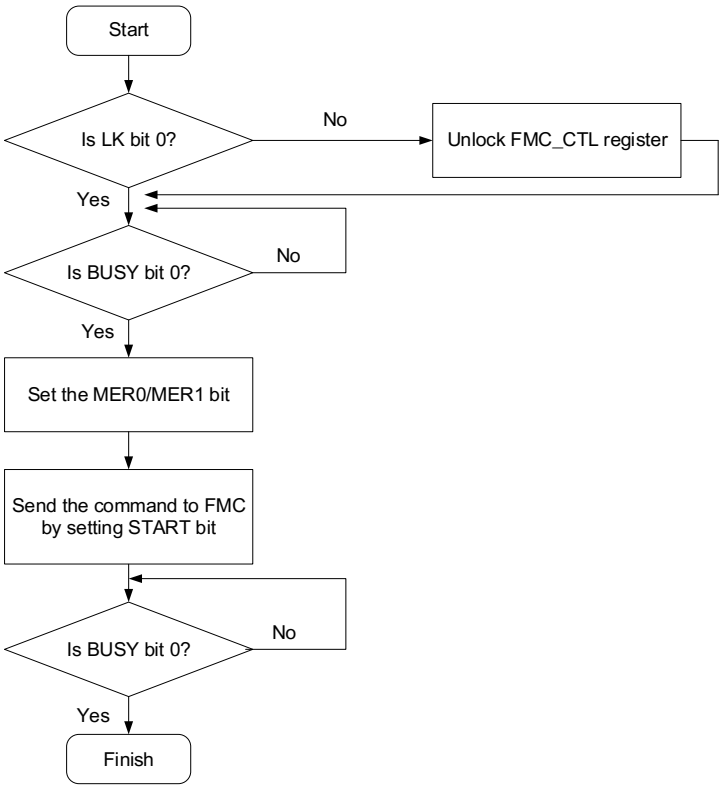
1. Unlock the FMC_CTL register if necessary.
2. Check the BUSY bit in FMC_STAT register to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
3. Set MER0 bit in FMC_CTL register if erase Bank0 only. Set MER1 bit in FMC_CTL register if erase Bank1(include Bnak1_Ex) and only. Set MER0/MER1 bits in FMC_CTL register if erase entire flash.
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 using a CBUS access.

Note: Do not power off or reset during the erase process.

When ENDIE bit in the FMC_CTL register is set and the mass erase operation is executed successfully, the END in FMC_STAT register is set, and an interrupt will be triggered. Since all flash 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 mass erase operation indicates the mass erase operation flow.

Figure 3-3. Process of mass erase operation



3.3.8. Main flash programming

The FMC provides a 64-bit double word / 32-bit word / 16-bit half word / 8-bit byte programming function which is used to modify the main flash memory contents. Programming 64-bit alignment area multiple times will reduce the security of ECC. For example, program 0x08000000(64-bit alignment address) 32-bit firstly, then it is not safe to program 0x08000004 32-bit. It is safe to program 0x08000008 32-bit. Therefore, it is highly recommennded to use 64-bit double word programming to ensure the security of ECC.

The following steps show the register access sequence of the word programming 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 PG bit in FMC_CTL register.
4. Write the data to be programed by CBUS with desired absolute address (0x08XX XXXX).

If CBUS program is 32-bit and the DWPGE bit is set to 1(64-bit program to flash memory), the CBUS write lower 32-bit firstly and then higher 32-bit to form a 64-bit data. The 64-bit data is programmed to flash memory. The data to be programed must be double-word aligned.

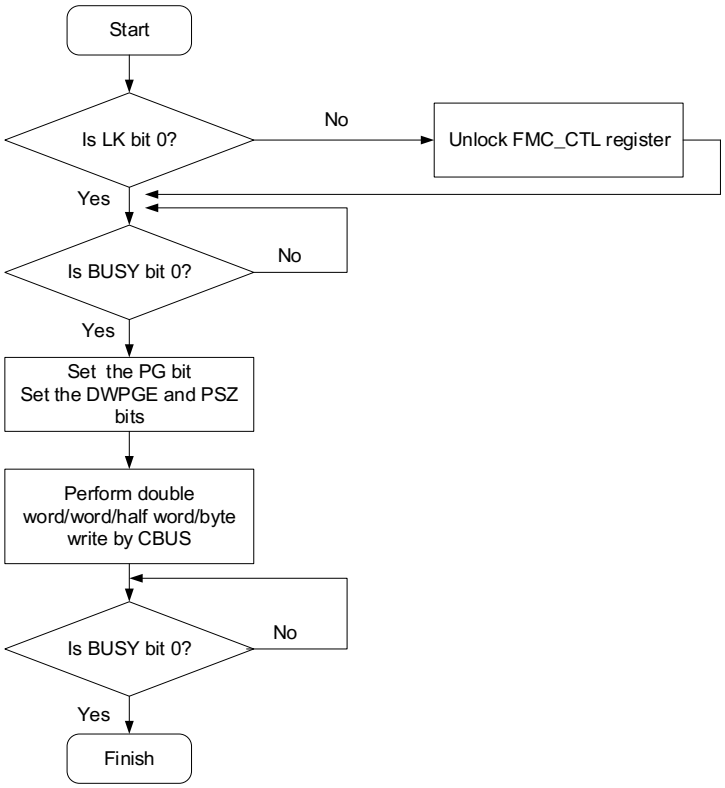
Write a 32-bit word/16-bit half word/8-bit byte (must match PSZ bits in FMC_CTL register and DWPGE=0) to desired absolute address by CBUS.

Note: Multiple writes will reduce ECC security. Therefore, it is recommennded to write only one time.

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 using a CBUS access.

When the operation is executed successfully, the END 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 double word/word/half word/byte programming operation must match DWPGE and PSZ bits in FMC_CTL register. If not match, PGMERR bit in the FMC_STAT register will be set. Note that the PG bit must be set before the word/half word/byte programming operation, or else PGSERR bit in the FMC_STAT register will be set. Additionally, the program operation will be ignored on erase/program protected sectors and WPERR bit in FMC_STAT is set. If data or address is not aligned in 64bit program, PGAERR bit in FMC_STAT register is set. In these conditions, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC_CTL register is set. The software can check the PGMERR bit, PGSERR or WPERR bit in the FMC_STAT register to detect which condition occurred in the interrupt handler. ***Figure 3-4. Proccess of word program operation*** displays the word programming operation flow.

Figure 3-4. Proccess of word program operation



Note: 1. Reading the flash should be avoided when a program/erase operation is ongoing in the same bank. And flash memory accesses failed if the CPU enters the power saving modes.
2. Do not power off or reset during the programming process.

3.3.9. OTP block programming

The FMC provides a 64bit double word / 8-bit byte programming function which is used to modify OTP0 / OTP1 / OTP2 contents. Additionally, 32-bit word / 16-bit half word programming can be used to modify OTP1 data blocks / OTP2 data blocks. The programming sequence is same as main flash programming. All OTP can only be programmed one time and not be erased. Each byte of lock blocks can only be programmed one time from 0xFF to 0x00, no other value.

The OTP0 block can be divided to 64 data blocks which has 1 byte each and 1 lock block which has 64 bytes. The lock block address is from 0x1FFF 7840 to 0x1FFF 787F. The data block address is from 0x1FFF 7800 to 0x1FFF 783F. Each lock byte (0x00: lock 0xFF:no lock) can lock corresponding data blocks to prevent program to this data block. The lock byte 0 on 0x1FFF 7840 locks data block 0 on 0x1FFF 7800. The lock byte 1 on 0x1FFF 7841 locks data block 1 on 0x1FFF 7801, and so on.

Table 3-5. OTP0 lock

Lock byte	Lock byte addr	Locked data block	Lockeddata address
0	0x1FFF 7840	0	0x1FFF 7800
1	0x1FFF 7841	1	0x1FFF 7801

Lock byte	Lock byte addr	Locked data block	Lockeddata address
. . .	. . .	. . .	. . .
62	0x1FFF 787E	62	0x1FFF 783E
63	0x1FFF 787F	63	0x1FFF 783F

The OTP1 block can be divided to 16 data blocks which has 8K bytes each and 1 lock block which has 16 bytes. The lock block address is from 0x1FF2 0200 to 0x1FF2 020F. The data block address is from 0x1FF0 0000 to 0x1FF1 FFFF. Each lock byte (0x00: lock 0xFF:no lock) can lock corresponding data blocks to prevent program to this data block. The lock byte 0 on 0x1FF2 0200 locks data block 0 on 0x1FF0 0000. The lock byte 1 on 0x1FF2 0201 locks data block 1 on 0x1FF0 2000, and so on. OTP1 data blocks can be read or not determined by OTP1REN[15:0] in FMC_OPT1CFG register. Reading a locked block of data will cause a bus error.

Table 3-6. OTP1 lock

Lock byte	Lock byte addr	Locked data block	Lockeddata address
0	0x1FF2 0200	0	0x1FF0 0000 - 0x1FF0 1FFF
1	0x1FF2 0201	1	0x1FF0 2000 - 0x1FF0 3FFF
. . .	. . .	. . .	. . .
14	0x1FF2 020E	14	0x1FF1 C000 - 0x1FF1 DFFF
15	0x1FF2 020F	15	0x1FF1 E000 - 0x1FF1 FFFF

The OTP2 block can be divided to 16 data blocks which has 32 bytes each and 1 lock block which has 32 bytes. The lock block address is from 0x1FF2 0210 to 0x1FF2 022F. The data block address is from 0x1FF2 0000 to 0x1FF2 01FF.

The OTP2 write lock block address is from 0x1FF2 0210 to 0x1FF2 021F. Each lock byte (0x00: lock 0xFF:no lock) can lock corresponding data blocks to prevent program to this data block. The lock byte 0 on0x1FF2 0210 locks data block 0, and so on.

The OTP2 read lock block address is from 0x1FF2 0220 to 0x1FF2 022F. Each lock byte (0x00: lock 0xFF: no lock) can lock corresponding data blocks to prevent read access. The lock byte 16 on 0x1FF2 0220 locks data block 0, and so on. When the RLBE bit in FMC_CTL is set, the OTP2 data block corresponding to the read lock block cannot be read. For example, security verification data is stored in OTP2, and the security startup program starting from OTP1 can read OTP2 information for verification. After the verification is completed, the RLBE is set and then jumps to other programs. the OTP2 data block corresponding to the read lock block cannot be read until the next reset.

Table 3-7. OTP2 lock

Write lock byte	Write lock byte addr	Read lock byte	Read lock byte addr	Locked data block	Lockeddata address
0	0x1FF2 0210	16	0x1FF2 0220	0	0x1FF2 0000 - 0x1FF2 001F

Write lock byte	Write lock byte addr	Read lock byte	Read lock byte addr	Locked data block	Lockeddata address
1	0x1FF2 0211	17	0x1FF2 0221	1	0x1FF2 0020 - 0x1FF2 003F
. . .	. . .	. . .	. . .	. . .	. . .
15	0x1FF2 021F	31	0x1FF2 022F	15	0x1FF2 01E0 - 0x1FF2 01FF

3.3.10. Option bytes modify

The FMC provides an erase and then program function which is used to modify the option bytes block in flash. The following steps show the erase sequence.

1. Unlock the FMC_OBCTLx 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 option byte value in FMC_OBCTL0 and FMC_OBCTL1 registers.
4. Send the option bytes erase command to the FMC by setting the OB_START bit in FMC_OBCTL0 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.

When the operation is executed successfully, the END 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: During the modification process, power-off or reset is prohibited; otherwise, it may cause option byte abnormalities, resulting in program failure to run. Code and option bytes need to be reprogrammed.

3.3.11. Option bytes description

The option bytes block is reloaded to FMC_OBCTL0 and FMC_OBCTL1 registers after each system reset, and the option bytes take effect. ***Table 3-8. Option byte*** is the detail of option bytes.

Table 3-8. Option byte

Address	Name	Description
0x1FFF C000	USER	[7]: nRST_STDBY 0: generates a reset instead of entering standby mode 1: no reset when entering standby mode (Factory value) [6]: nRST_DPSLP 0: generates a reset instead of entering Deep-sleep mode 1: no reset when entering Deep-sleep mode (Factory value) [5]: nWDG_HW 0: hardware free watchdog

Address	Name	Description
		1: software free watchdog (Factory value) [4]: BB 0: boot from bank0, when configured boot from main memory (Factory value) 1: boot from bank1 or bank0 if bank1 is void, when configured boot from main memory. Set NWA bit in this case will improve software performance. [3:2]: BOR_TH (Brown out reset threshold) 00: BOR threshold value 3 01: BOR threshold value 2 10: BOR threshold value 1 11: NO BOR function (Factory value) [1]: ECCEN 0: disable ECC 1: enable ECC (Factory value) **Note**: This bit is only valid after power reset. If ECCEN is set from 0 to 1, backup SRAM must be rewriten before use it. [0]: Reserved
0x1FFF C001	SPC	Security Protection Code 0xAA: No protection (Factory value) any value except 0xAA or 0xCC: Protection level low 0xCC: Protection level high
0x1FFF C008	WP0	[7:0]: WP0[7:0] Sector Erase/Program Protection bit 7 to 0 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1 (Factory value).
0x1FFF C009	WP0	[7]: DRP CBUS data read protection bit. 0: The WP0 bits used as erase/program protection of each sector (Factory value). 1: The WP0 bits used as erase/program protection and CBUS read protection of each sector [6]: Reserved [5]: NWA No waiting time area select 0: Bank1 1: Bank0 (Factory value) **Note**: This bit is only valid after power reset. This bit is only valid for 4MB dual bank series. [4]: Reserved

Address	Name	Description
		[3:0]: WP0[11:8] 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.
0x1FFF C00C	WP0	[7:0]: WP0[19:12] 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.
0x1FFE C008	WP1	[7:0]: WP1[7:0] Sector Erase/Program Protection bit 7 to 0 for Bank1 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.
0x1FFE C009	WP1	[7:4]: Reserved [3:0]: WP1[11:8] Sector Erase/Program Protection bit 11 to 8 for Bank1 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1
0x1FFE C00C	WP1	[7:0]: WP1[19:12] Sector Erase/Program Protection bit 12 to 19 for Bank1 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1

3.3.12. 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 MER0/MER1 set or SN not valid. If the WPERR bit is set and the ERRIE bit is also set to 1 to enable the corresponding interrupt, then the Flash operation error interrupt will be triggered by the FMC to draw the attention of the CPU. The sector protection function can be individually enabled by configuring the WP0[19:0] / WP1[19:0] bit field to 0 when DRP is 0 or to 1 when DRP is 1 in the option bytes.

Table 3-9. WP0/WP1 bit for sectors protected

WP0/WP1 bit	sectors protected
WP0[0]	Sector0
WP0[1]	Sector1
WP0[2]	Sector2
…	…
WP0[18]	Sector18
WP0[19]	Sector19
WP1[0]	Sector20
WP1[1]	Sector21
WP1[2]	Sector22
…	…
WP1[18]	Sector38
WP1[19]	Sector39~Sector53

3.3.13. CBUS read protection

The FMC provides CBUS data read protection functions to prevent CBUS read operations on corresponding sector when DRP set to 1. The CBUS read will not be accepted by the FMC on protected sectors. If the CBUS read command is sent to the FMC on a protected sector, the RDCERR bit in the FMC_STAT register will then be set by the FMC. If the RDCERR bit is set and the ERRIE bit is also set to 1 to enable the corresponding interrupt, then the Flash operation error interrupt will be triggered by the FMC to draw the attention of the CPU. The sector protection function can be individually enabled by configuring the WP0 [19:0]/WP1 [19:0] bit field to 1 and set DRP to 1 in the option bytes.

If the DRP is 1, modify DRP to 0 or change WP0 [19:0]/WP1 [19:0] bit field from 1 to 0 must performed during changing the security protection level low to no security protection. Otherwise, the option byte modification ignored and WPERR bit in the FMC_STAT register will then be set by the FMC. If the WPERR bit is set and the ERRIE bit is also set to 1 to enable the corresponding interrupt, then the Flash operation error interrupt will be triggered by the FMC to draw the attention of the CPU.

3.3.14. 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/firmware from illegal users. ***Table 3-10. Security protection*** shows different configurations. There are 3 levels for protecting:

No protection: when setting EFSPC=0 in EFUSE control and SPC byte to 0xAA, no protection performed. The main flash and option bytes block are accessible by all operations.

Protection level low: when setting EFSPC=1 in EFUSE control or SPC byte 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 is forbidden. If a read operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, a 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 mass erase for main flash will be performed.

Note: User should not perform other operations (e.g. reset) before the mass erase is finished.

Protection level high: when setting SPC byte 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 option bytes cannot be reprogrammed. So, if protection level high is programmed, it cannot move back to protection level low or no protection level.

Table 3-10. Security protection

EFSPC	SPC[7:0]	Security Protection
0	0xAA	No protection
1/0	except 0xAA or 0xCC	Protection level low
1	except 0xCC	Protection level low
1/0	0xCC	Protection level high

3.3.15. EFUSE macro description

The efuse macro stores 2 system parameters.

Table 3-11. system parameters shows the details of each efuse byte.

Table 3-11. system parameters

Parameter	Width /B	Start address	Program-protected	Read-protected	Description	Note
EFUSE control	1B	1	Can write multiple times, but can not rollback	Read out and take effect after system reset and keep unchanged, bus readable	Control bytes of relevant parameters required for MCU startup. For more details, refer to *EFUSE control register (EFUSE_CTL)*	User
User data	1B	2	Can write multiple times, but can not rollback	Read out after system reset, bus readable	User defined data For more details, refer to *EFUSE user data register (EFUSE_USER_DATA)*	User

Note: System parameters must be read by its size. And it is also recommended to write according to its size too. The value in efuse macro will be loaded after system reset.

3.3.16. EFUSE read operation

The value of the efuse can only be accessed through the corresponding register. After system reset, the efuse value take effect and reloaded to corresponding register. When read the EFUSE control and User data related bytes in the efuse, user need to follow the following steps:

1. Make sure system clock comes directly from IRC16M and the V_{CORE} voltage is 1.1V.
2. Clear the RDIF bit in EFUSE_CS and make sure there are no overstep boundary errors.
3. Clear the EFRW bit in EFUSE_CS.
4. Write the desired efuse address and size to EFUSE_ADDR register.
5. Set the EFSTR bit EFUSE_CS register.
6. Wait until the reading operation has been finished by checking the RDIF bit in EFUSE_CS register.
7. Read the register value corresponding to the efuse.

When the read operation is executed successfully, the RDIF in EFUSE_CS register is set, and an interrupt will be triggered by EFUSE if the RDIE bit in the EFUSE_CS register is set.

3.3.17. EFUSE 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. Make sure system clock comes directly from IRC16M and the V_{CORE} voltage is 1.1V.
2. Clear the PGIF bit if it is SET, and make sure there are no overstep boundary errors.
3. SET the EFRW bit in EFUSE_CS.
4. Write the desired efuse address and size to EFUSE_ADDR register.
5. Write the data to the corresponding register.
6. Set the EFSTR bit EFUSE_CS register.
7. Wait until the writing operation has been finished by checking the PGIF bit in EFUSE_CS register.

When the write operation is executed successfully, the PGIF in EFUSE_CS register is set, and an interrupt will be triggered by EFUSE if the PGIE bit in the EFUSE_CS 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, OVBERIF bit in the EFUSE_CS register will be set, and an interrupt will be triggered by EFUSE if the OVBERIE bit in the EFUSE_CS register is set.

3.4. Register definition

FMC base address: 0x4002 3C00

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).

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

Bits	Fields	Descriptions
31:0	KEY[31:0]	FMC_CTL unlock register These bits can only be written by software. 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).

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

Bits	Fields	Descriptions
31:0	OBKEY[31:0]	FMC_OBCTLx option byte operation unlock register These bits are only can be written by software Write OBKEY[31:0] with keys to unlock option byte command in FMC_OBCTLx register.

3.4.3. Status register (FMC_STAT)

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															BUSY
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							RDCERR	PGSERR	PGMERR	PGAERR	WPERR	Reserved	LDECCDET	OPERR	END
							rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	BUSY	The flash is busy bit. 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.
15:9	Reserved	Must be kept at reset value.
8	RDCERR	Read CBUS protection error flag bit. This bit is set by hardware when a CBUS read to the sector which is a CBUS read protection sector. The software can clear it by writing 1.
7	PGSERR	Program sequence error flag bit. This bit is set by hardware when program to flash while the PG bit in FMC_CTL registers is not set. The software can clear it by writing 1.
6	PGMERR	Program size not match error flag bit. This bit is set by hardware when program write size (byte/half-word/word access) does not match the PSZ bits in FMC_CTL registers. The software can clear it by writing 1.
5	PGAERR	Program alignment error flag bit This bit is set by hardware when write data or address is not alignment in double word program. The software can clear it by writing 1.
4	WPERR	Erase/Program protection error flag bit. When erase/program on protected pages, this bit is set by hardware. The software can clear it by writing 1.
3	Reserved	Must be kept at reset value.
2	LDECCDET	Two bits ECC error flag bit when load code. When detected ECC two bits error, this bit is set by hardware. The software can clear it by writing 1.
1	OPERR	Flash operation error flag bit.

		This bit is set by hardware when an error (an error while sets RDCERR/PGSERR/PGMERR/WPERR bit) occurs on a flash operation and ERRIE bit in FMC_CTL register is set. The software can clear it by writing 1.
0	END	End of operation flag bit. When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1.

3.4.4. Control register (FMC_CTL)

Address offset: 0x10

Reset value: 0x8000 0000

This register 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	RLBE	NWLDE	Reserved		LDECCIE	ERRIE	ENDIE	Reserved							START
rs	w	rw			rw	rw	rw								rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER1	Reserved			SN[5]	DWPGE	PSZ[1:0]		SN[4:0]					MER0	SER	PG
rw				rw	rw	rw		rw					rw	rw	rw

Bits	Fields	Descriptions
31	LK	FMC_CTL lock bit This bit is cleared by hardware when right sequence written to FMC_KEY register. This bit can be set by software.
30	RLBE	Enable read lock block for OTP2. This bit can only be set to 1 to enable read lock block for OTP2. Once the software sets the value to 1, the data block corresponding to the read lock block cannot be read. Reset value is restored after reset.
29	NWLDE	Enable no waiting time area load when system reset. This bit is only restored by power on reset. 0: don't copy flash content to buffer memory when system reset. 1: copy flash content to buffer memory when system reset. Don't support in JTAG debug mode.
28:27	Reserved	Must be kept at reset value.
26	LDECCIE	Load code ECC error interrupt enable bit. This bit is set or cleared by software. 0: no interrupt generated by hardware 1: error interrupt enable
25	ERRIE	Error interrupt enable bit.

		This bit is set or cleared by software. 0: no interrupt generated by hardware 1: error interrupt enable
24	ENDIE	End of operation interrupt enable bit. This bit is set or cleared by software. 0: no interrupt generated by hardware 1: end of operation interrupt enable
23:17	Reserved	Must be kept at reset value.
16	START	send erase command to FMC bit. This bit is set by software to send erase command to FMC. This bit is cleared by hardware when the BUSY bit is cleared.
15	MER1	main flash mass erase for bank1command bit. This bit is set or cleared by software. 0: no effect 1: main flash mass erase command for bank1
14:12	Reserved	Must be kept at reset value.
11	SN[5]	Refer to SN[4:0].
10	DWPGE	Double word program size bit enable. 0: Refer to PSZ[1:0] 1: Double word Program size
9:8	PSZ[1:0]	Program size bit when DWPGE=0. The bits are set or cleared by software. 00: Program by byte access 01: Program by half-word access 10/11: Program by word access
7:3	SN[4:0]	Select which sector number to be erased. SN[5:0] 000000: select sector 0 000001: select sector 1 ... 110100: select sector 52 110101 select sector 53 110110 - 111111: Reserved
2	MER0	main flash mass erase for bank0 command bit. This bit is set or cleared by software. 0: no effect 1: main flash mass erase command for bank0
1	SER	main flash sector erase command bit. This bit is set or cleared by software.

		0: no effect 1: main flash sector erase command
0	PG	main flash program command bit. This bit is set or cleared by software. 0: no effect 1: main flash program command

NOTE: This register should be reset after the corresponding flash operation completed.

3.4.5. Option byte control register 0 (FMC_OBCTL0)

Address offset: 0x14

Reset value: 0xXXXX XXXX, the initial value is 0x3FFF AAED. 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
DRP	Reserved	NWA	ECCEN	WP0[11:0]											
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	nWDG_HW	BB	BOR_TH[1:0]		OB_START	OB_LK
rw								rw	rw	rw	rw	rw		rs	rs

Bits	Fields	Descriptions
31	DRP	CBUS data read protection bit. 0: The WPx bits used as erase/program protection of each sector 1: The WPx bits used as erase/program protection and CBUS read protection of each sector.
30	Reserved	Must be kept at reset value.
29	NWA	Select no waiting time area. 0: Bank1 1: Bank0
28	ECCEN	ECC enable bit. 0: disable ECC. 1: enable ECC.
27:16	WP0[11:0]	Erase/program protection of each sector when DRP is 0. Erase/program protection and CBUS read protection of each sector when DRP is 1. WP0[0] affect sector 0, WP0[1] affect sector 1, etc. 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.
15:8	SPC[7:0]	option byte security protection code.

		0xAA: no security protection 0xCC: security protection level high any value except 0xAA or 0xCC : security protection level low.
7	nRST_STDBY	option byte standby reset value. 0: generates a reset instead of entering standby mode 1: no reset when entering standby mode
6	nRST_DPSLP	option byte deepsleep reset value. 0: generates a reset instead of entering Deep-sleep mode 1: no reset when entering Deep-sleep mode
5	nWDG_HW	option byte watchdog value. If change this bit, a system reset needed to take effect. 0: hardware free watchdog 1: software free watchdog
4	BB	option byte boot bank value. 0: boot from bank0, when configured boot from main memory 1: boot from bank1 or bank0 if bank1 is void, (or Bootloader continues executing if bank1 and bank0 are both void, and the chip is not under security protection level high,) when configured boot from main memory.
3:2	BOR_TH[1:0]	option byte BOR threshold value. 00: BOR threshold value 3 01: BOR threshold value 2 10: BOR threshold value 1 11: No BOR function.
1	OB_START	send option byte change command to FMC bit. This bit is set by software to send option byte change command to FMC. This bit is cleared by hardware when the BUSY bit is cleared.
0	OB_LK	FMC_OBCTLx lock bit This bit is cleared by hardware when right sequence written to FMC_OBKEY register. This bit can be set by software.

3.4.6. Option byte control register 1 (FMC_OBCTL1)

Address offset: 0x18

Reset value: 0xXXXX XXXX. the initial value is 0x0FFF FFFF. 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
Reserved				WP1[11:0]											
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

WP1[19:12]	WP0[19:12]
rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:16	WP1[11:0]	Erase/program protection of each sector when DRP is 0. Erase/program protection and CBUS read protection of each sector when DRP is 1. WP1[0] affect sector 20, WP1[1] affect sector 21, etc. 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.
15:8	WP1[19:12]	Erase/program protection of each sector when DRP is 0. Erase/program protection and CBUS read protection of each sector when DRP is 1. WP1[12] affect sector 32, WP1[13] affect sector 33, etc. Exceptional, WP1[19] affect sector 39~53. 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.
7:0	WP0[19:12]	Erase/program protection of each sector when DRP is 0. Erase/program protection and CBUS read protection of each sector when DRP is 1. WP0[0] affect sector 0, WP0[1] affect sector 1, etc. 0: Erase/program protection when DRP is 0. No effect when DRP is 1. 1: No effect when DRP is 0. Erase/program protection and CBUS read protection when DRP is 1.

3.4.7. Page erase configuration register (FMC_PECFG)

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
PE_EN	Reserved							PE_ADDR[28:16]							
rw								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE_ADDR[15:0]															
rw															

Bits	Fields	Descriptions
31	PE_EN	The enable bit of page erase function 0: Disable page erase. 1: Enable page erase.

30:29	Reserved	Must be kept at reset value.
28:0	PE_ADDR[28:0]	Page address (4KB alignment).

3.4.8. Unlock page erase key register (FMC_PEKEY)

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
PE_KEY[31:16]															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE_KEY[15:0]															
w															

Bits	Fields	Descriptions
31:0	PE_KEY[31:0]	FMC_PECFG unlock register These bits can only be written by software. Write key value 0xA9B8C7D6 into KEY[31:0] to unlock FMC_PECFG register.

3.4.9. OTP1 configuration register (FMC_OTP1CFG)

Address offset: 0x28

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
OPT1REN[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	OTP1REN[15:0]	OTP1 read enable. OTP1REN[x] decides OTP1 data block x read or not, x=0..15. 0: data block can not be read 1: data block can be read Software can write 0, but only reset can set to 1.

3.4.10. Load code ECC error address0 (FMC_LDECCADDR0)

Address offset: 0x2C

Reset value: 0xFFFF FFFF. Load Flash values after reset. If no load code ECC error, the default value is 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
LDECCADDR0[31:16]															
rs_r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDECCADDR0[15:0]															
rs_r															

Bits	Fields	Descriptions
31:0	LDECCADDR0[31:0]	ECC two bits error base address (aligned to 64bit) when load code from main flash / bootloader OTP1. Any bits in 64bit may be wrong.

3.4.11. Load code ECC error address1 (FMC_LDECCADDR1)

Address offset: 0x30

Reset value: 0xFFFF FFFF. Load Flash values after reset. If no load code ECC error, the default value is 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
LDECCADDR1[31:16]															
rs_r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDECCADDR1[15:0]															
rs_r															

Bits	Fields	Descriptions
31:0	LDECCADDR1[31:0]	ECC two bits error base address (aligned to 64bit) when load code from main flash / bootloader OTP1. Any bits in 64bit may be wrong.

3.4.12. Load code ECC error address2 (FMC_LDECCADDR2)

Address offset: 0x34

Reset value: 0xFFFF FFFF. Load Flash values after reset. If no load code ECC error, the default value is 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
LDECCADDR2[31:16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDECCADDR2[15:0]															
rs_r															

Bits	Fields	Descriptions
31:0	LDECCADDR2[31:0]	ECC two bits error base address (aligned to 64bit) when load code from main flash / bootloader OTP1. Any bits in 64bit may be wrong.

3.4.13. Option bytes status register (FMC_OBSTAT)

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													SPCH	SPCL	Reserved
													r	r	

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	SPCH	Security protection is level high now.
1	SPCL	Security protection is level low now.
0	Reserved	Must be kept at reset value.

3.4.14. Product ID register (FMC_PID)

Address offset: 0x100

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

These bits are read only by software.

These bits are unchanged constant after power on. These bits are one time program when the chip produced.

3.4.15. EFUSE control and status register (EFUSE_CS)

Address offset: 0x200

Reset value: 0x0000 0000

This register 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					OVBERIC	RDIC	PGIC	Reserved	OVBERIE	RDIE	PGIE	Reserved	OVBERIF	RDIF	PGIF
					rc_w1	rc_w1	rc_w1		rw	rw	rw		r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													EFBYP	EFRW	EFSTR
													rw	rw	rw

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	OVBERIC	Clear bit for overstep boundary error interrupt flag. 0: No effect 1: Clear error flag
25	RDIC	Clear bit for read operation completed interrupt flag. 0: No effect 1: Clear read operation completed interrupt flag
24	PGIC	Clear bit for program operation completed interrupt flag. 0: No effect 1: Clear program operation completed interrupt flag
23	Reserved	Must be kept at reset value.
22	OVBERIE	Enable bit for overstep boundary error interrupt. 0: Disable the overstep boundary error interrupt 1: Enable the overstep boundary error interrupt
21	RDIE	Enable bit for read operation completed interrupt. 0: Disable the read operation completed interrupt 1: Enable the read operation completed interrupt
20	PGIE	Enable bit for program operation completed interrupt. 0: Disable the program operation completed interrupt 1: Enable the program operation completed interrupt
19	Reserved	Must be kept at reset value.
18	OVBERIF	Overstep boundary error flag.

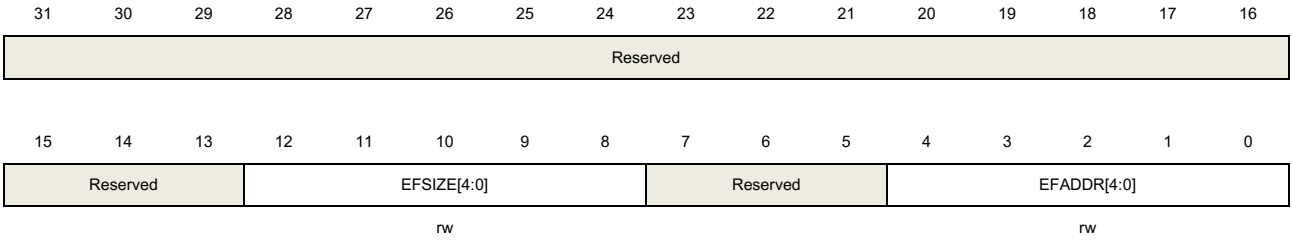
		0: No overstep boundary error occurred 1: Overstep boundary error has occurred
17	RDIF	Read operation complete flag. 0: Read efuse operation not completed 1: Read efuse operation completed
16	PGIF	Program operation completed flag. 0: Program efuse operation not completed 1: Program efuse operation completed
15:3	Reserved	Must be kept at reset value.
2	EFBYP	EFUSE LDO bypass. This bit can be set only when EFSTR is 0. And users should make sure the external voltage supply is stable before seting EFSTR if set this bit. The program voltage should whitin 2.25V~2.75V. And typical voltage is 2.5V. 0: EFUSE program use internal LDO supply. 1: EFUSE program use external supply from PAD VEFUSE.
1	EFRW	The selection of efuse operation. 0: Read efuse 1: Write efuse This bit cannot be modified when the EFSTR bit is 1.
0	EFSTR	Start efuse operation. This bit is set by software and cleared by hardware. 0: No effect 1: Start read or write efuse operation

3.4.16. EFUSE address register (EFUSE_ADDR)

Address offset: 0x204

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:8	EFSIZE[4:0]	Read or write efuse data size.
7:5	Reserved	Must be kept at reset value.

4:0	EFADDR[4:0]	Read or write efuse data start address.

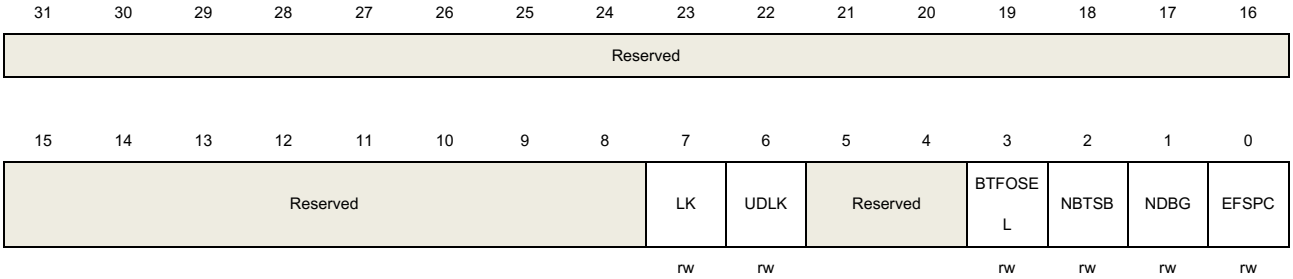
Note: This register cannot be modified when the EFSTR bit is 1.

3.4.17. EFUSE control register (EFUSE_CTL)

Address offset: 0x208

Reset value: 0x0000 0000

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



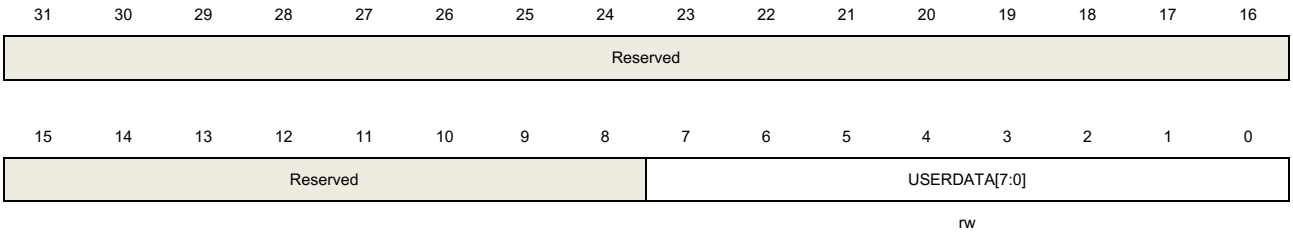
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	LK	EFUSE_CTL register lock bit. 0: unlock EFUSE_CTL register 1: lock EFUSE_CTL register
6	UDLK	EFUSE_USER_DATA register lock bit 0: unlock EFUSE_USER_DATA register 1: lock EFUSE_USER_DATA register
5:4	Reserved	Must be kept at reset value.
3	BTFOSEL	Select Boot from flash or OTP1. This bit actives when NBTSB is 1 or BOOT0 is 0. 0: boot from main flash block 1: boot from flash OTP1 block
2	NBTSB	Not boot from SRAM or bootloader. 0: can boot from SRAM or bootloader 1: can't boot from SRAM or bootloader
1	NDBG	Debugging permission setting. 0: no influence 1: Can not debug
0	EFSPC	EFUSE Security protection. Forbid SPC from no protect to level low. 0: no protect. 1: force chip SPC level low.

3.4.18. EFUSE user data register (EFUSE_USER_DATA)

Address offset: 0x20C

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	USERDATA[7:0]	EFUSE USER_DATA value.

4. Power management unit (PMU)

4.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32F527xx series. 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 GD32F527xx devices, there are three power domains, including V_{DD} / V_{DDA} domain, V_{CORE} domain, and Backup domain, as is shown in the ***Figure 4-1. Power supply overview***. The power of the V_{DD} domain is supplied directly by VDD. An embedded LDO in the V_{DD} / V_{DDA} domain 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 VDD supply is shut down.

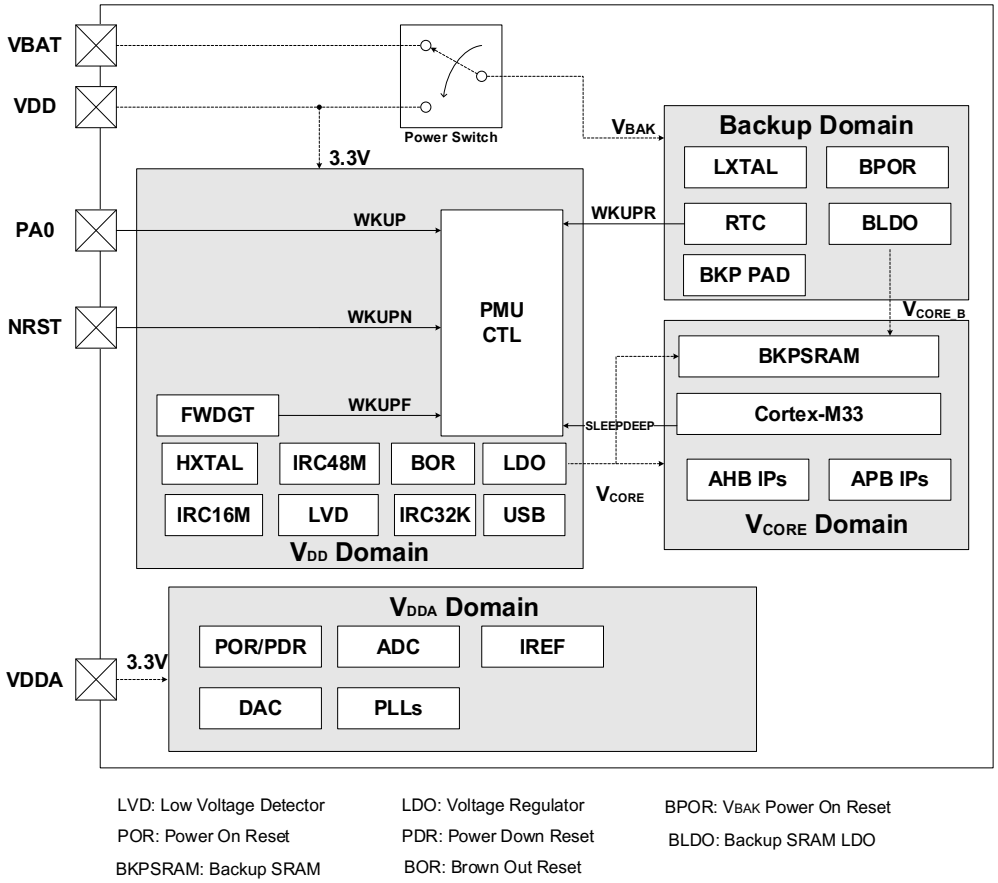
4.2. Characteristics

- 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 and a Backup LDO(BLDO) dedicate to Backup SRAM.
- Low Voltage Detector can issue an interrupt or event when the power is lower than a programmed threshold.
- Battery power (V_{BAT}) for Backup domain when VDD is shut down.
- Ultra power saving for low-driver mode in Deep-sleep mode. And high-driver mode for high frequency.
- 4K bytes backup SRAM powered by V_{CORE} which source from VDD or V_{BAK} for data protection of user application data when VDD shut down.
- LDO output voltage select for power saving.

4.3. Function overview

Figure 4-1. Power supply overview provides details on the internal configuration of the PMU and the relevant power domains.

Figure 4-1. Power supply overview



4.3.1. Battery backup domain

The Backup domain is powered by the VDD or the battery power source (VBAT) selected by the internal power switch, and the V_{BAK} pin which drives Backup domain, supplies power for RTC unit, LXTAL oscillator, BPOR and BLDO, and four BKP PAD including PC13 to PC15 and PI8. In order to ensure the content of the Backup domain registers and the RTC supply, when VDD supply is shut down, VBAT 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 VBAT pin externally to VDD pin with a 100nF external ceramic decoupling capacitor.

The Backup domain reset sources include 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 clock divided

by 2 to 31. When VDD is shut down, only LXTAL is valid for RTC. Before entering the power saving mode by executing the WFI / WFE instruction, the Cortex®-M33 can setup the RTC register with an expected wakeup time and enable the wakeup function to achieve the RTC wakeup event. After entering the power saving mode for a certain amount of time, the RTC 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 VDD (V_{BAK} is connected to V_{DD}), the following functions are available:

- PC13 and PI8 can be used as GPIO or RTC function pin described in the ***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 VBAT (V_{BAK} is connected to V_{BAT}), the following functions are available:

- PC13 and PI8 can be used as RTC function pin described in the ***Real time clock (RTC)***.
- PC14 and PC15 can be used as LXTAL Crystal oscillator pins only.

Note: Since PC13, PC14, PC15 and PI8 are supplied through the power switch, which can only be obtained by a small current, the speed of GPIOs PC13 to PC15 and PI8 should not exceed 2MHz when they are in output mode (maximum load: 30pF).

4.3.2. Backup SRAM

There is 4K bytes backup SRAM in V_{CORE} domain. The backup SRAM can maintain data in Standby mode or VDD shut down state when BLDOON bit is set in PMU_CS register. In these modes, the backup SRAM powered by BLDO which sources from V_{BAK}. In other mode (not in Standby mode and VDD is not shut down), the backup SRAM can be accessed by system bus as normal SRAM and be power by LDO which sources from VDD.

The backup SRAM can only be accessed by user code when FMC is in security protection level low mode to prevent illegal code / data access. When the FMC goes from security protection level low mode to no security protection mode, the backup SRAM will be erased and all data lost. The backup SRAM is not reset by BKPRST in RCU_BDCTL register.

4.3.3. V_{DD} / V_{DDA} power domain

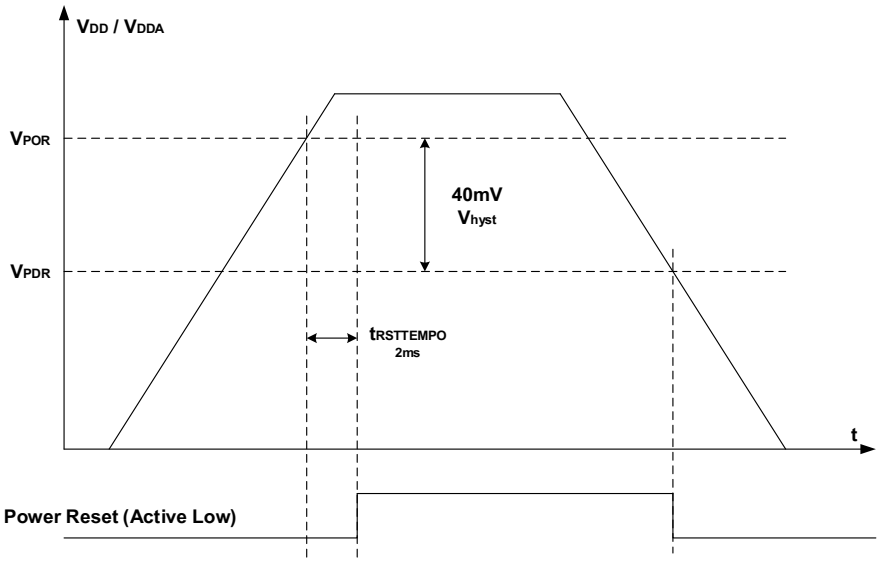
V_{DD} / V_{DDA} domain includes two parts: V_{DD} domain and V_{DDA} domain. V_{DD} domain includes HXTAL (high speed crystal oscillator), LDO (voltage regulator), POR / PDR (power on / down reset), FWDGT (free watchdog timer), all pads except PC13 / PC14 / PC15 / PI8, etc. V_{DDA} domain includes ADC / DAC (AD / DA converter), IRC16M (internal 16MHz RC oscillator), IRC48M (internal 48MHz RC oscillator), IRC32K (internal 32KHz RC oscillator), PLLs (phase locking loop), LVD (low voltage detector), etc.

VDD 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 three different status, including in the Sleep mode (full power on), 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_{DDA} 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 4-2. 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 V_{hyst} is hysteresis voltage.

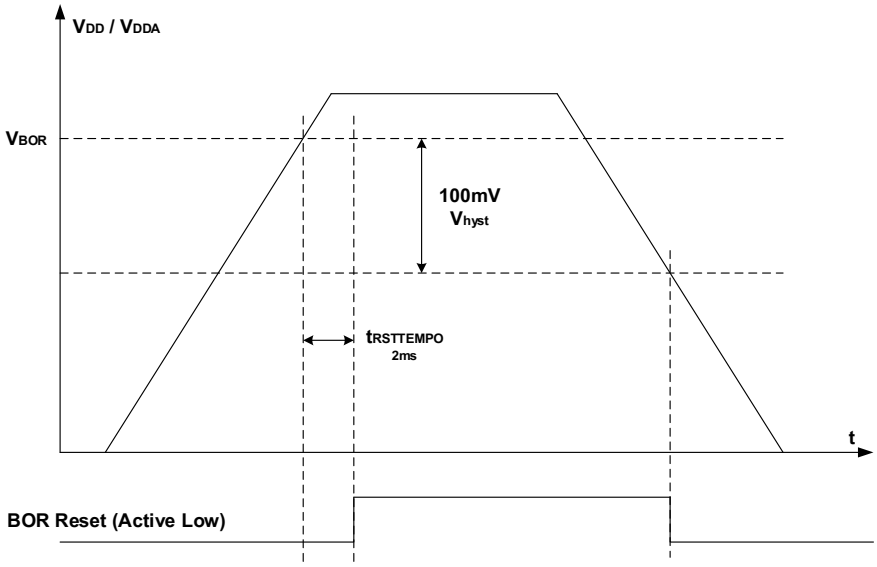
Figure 4-2. Waveform of the POR / PDR



Note: The PDR_ON pin is available on no less than 144-pin packages and BGA100-pins package. The internal POR / PDR circuit is enabled by the PDR_ON pin. To ensure the effectiveness of POR/PDR during the chip power-on and power-off stages, the PDR_ON pin needs to be kept at a high level. Refer to the recommended circuit diagram for the PDR_ON pin in the Datasheet.

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 0b11 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 no matter BOR_TH bits in option bytes are 0b11 or not. ***Figure 4-3. 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 V_{hyst} is hysteresis voltage.

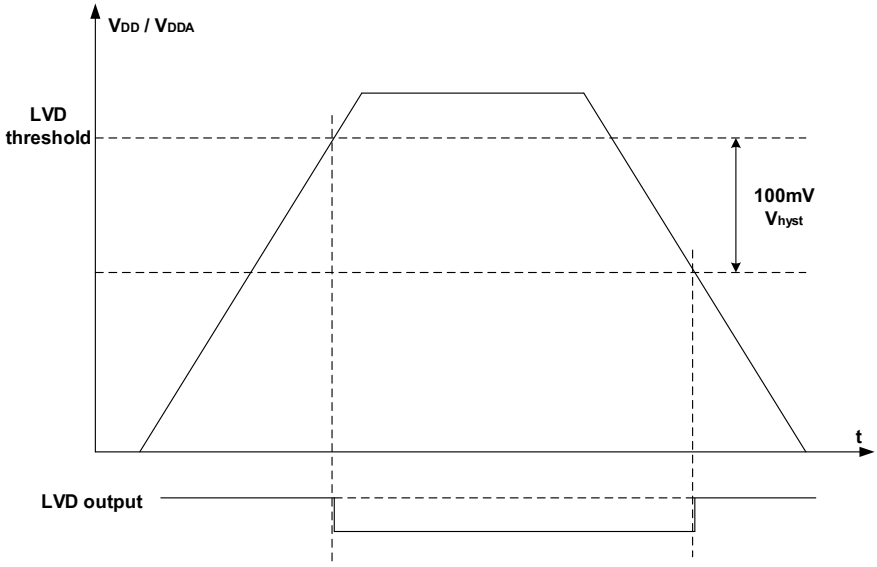
Figure 4-3. Waveform of the BOR



V_{DDA} domain

The LVD is used to detect whether the VDD supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU_CTL). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in the Power status register(PMU_CS), indicates if V_{DD} / V_{DDA} 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 4-4. 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 100mV.

Figure 4-4. Waveform of the LVD threshold



Generally, digital circuits are powered by VDD, while most of analog circuits are powered by VDDA. To improve the ADC and DAC conversion accuracy, the independent power supply VDDA is implemented to achieve better performance of analog circuits. VDDA can be externally connected to VDD through the external filtering circuit that avoids noise on VDDA, and VSSA 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 18-2. ADC input pins definition*** and ***Table 19-1. 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 BGA176-pins and BGA100-pins packages, or else the VREFN pin is not available and internally connected to VSSA.

4.3.4. V_{CORE} power domain

V_{CORE} power domain supplies power for Cortex®-M33 logic, AHB / APB peripherals, the APB interfaces for the Backup domain and the V_{DD} / V_{DDA} domain, etc. 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 which will be discussed in the following section. Refer to the datasheet for the V_{CORE} voltage value.

High-driver mode

If the V_{CORE} power domain runs with high frequency and opens many functions, it is recommended to enter high-driver mode. The following steps are needed when using high-driver mode.

- IRC16M or HXTAL must be selected as system clock.
- Set HDEN bit in PMU_CTL register to open high-driver mode.
- Wait until HDRF bit is 1 in PMU_CS register.
- Set HDS bit in PMU_CTL register to switch LDO to high-driver mode.
- Wait until HDSRF bit is 1 in PMU_CS register. And enter high-driver mode.
- Running the application at high frequency.

The high-driver mode can be exited by clearing the HDEN and HDS bits in PMU_CTL register after IRC16M or HXTAL is selected as system clock. The high-driver mode exits automaticly when exiting from Deep-sleep mode.

4.3.5. Power saving modes

After a system reset or a power reset, the GD32F527xx 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, PCLK2) or gating the clocks of the unused peripherals or configuring the LDO output voltage by LDOVS bits in PMU_CTL register. Besides, three power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode, and Standby mode.

Sleep mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex®-M33. In Sleep mode, only clock of Cortex®-M33 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex®-M33 System Control Register, and execute a WFI or 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 a WFE instruction, any wakeup event can wake up the system (If SEVONPEND is 1, any interrupt can wake up the system, refer to Cortex®-M33 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®-M33 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 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®-M33. In Deep-sleep mode, all clocks in the V_{CORE} domain are off, and all of IRC16M, IRC48M, HXTAL and PLLs are disabled. The contents of SRAM and registers are preserved. The LDO can operate normally or in low power mode depending on the LDOLP bit in the PMU_CTL register. Before entering the Deep-sleep mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M33 System Control Register, and clear the STBMOD bit in the PMU_CTL 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 1, any interrupt from EXTI lines can wake up the system, refer to Cortex®-M33 Technical Reference Manual). When exiting the Deep-sleep mode, the IRC16M is selected as the system clock. Notice that an additional wakeup delay will be incurred if the LDO operates in low power mode.

The LDEN, LDNP, LDLP and LDOLP bits in PMU_CTL register must be configured to enter the low-driver mode in Deep-sleep mode. The low-driver mode provides lower drive capability,

and the low-driver mode needs lower power.

Normal-driver / Normal-power: When configuring the LDEN bits in PMU_CTL register as 00, the Deep-sleep mode works in normal-driver mode. The LDO works in normal-power mode when the LDOLP bit in PMU_CTL register is cleared.

Normal-driver / Low-power: When configuring the LDEN bits in PMU_CTL register as 00, the Deep-sleep mode works in normal-driver mode. The LDO works in low-power mode when the LDOLP bit in PMU_CTL register is set.

Low-driver / Normal-power: When configuring the LDEN bits in PMU_CTL register as 11, and setting the LDNP bit, it enters the low-driver mode in Deep-sleep mode. The LDO works in normal-power mode when the LDOLP bit in PMU_CTL register is cleared.

Low-driver / Low-power: When configuring the LDEN bits in PMU_CTL register as 11, and setting the LDLP bit, it enters the low-driver mode in Deep-sleep mode. The LDO works in low-power mode when the LDOLP bit in PMU_CTL register is set.

No Low-driver: The Deep-sleep mode is not in low-driver mode by configuring LDEN to 00 in the PMU_CTL register.

Note: In order to enter Deep-sleep mode smoothly, all EXTI line pending status (in the EXTI_PD register) and related peripheral flags must be reset, refer to ***Table 7-3. EXTI source***. If not, the program will skip the entry process of Deep-sleep mode to continue to execute the following procedure.

Standby mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex®-M33, too. In Standby mode, the whole V_{CORE} domain is power off, the LDO is shut down, and all of IRC16M, HXTAL and PLL are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M33 System Control Register, and set the STBMOD bit in the PMU_CTL 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 four wakeup sources for the Standby mode, including the external reset from NRST pin, the RTC wakeup event including RTC Tamper event, RTC alarm event, RTC Time Stamp event or RTC Wakeup, the FWDGT reset, and the rising edge on WKUP pin. 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 (except Backup SRAM when BLDOON bit set) are lost in Standby mode. When exiting from the Standby mode, a power-on reset of V_{CORE} domain occurs and the Cortex®-M33 will execute instruction code from the 0x00000000 address.

Table 4-1. Power saving mode summary

Mode	Sleep	Deep-sleep	Standby
Description	Only CPU clock is off	1. All clocks in the V_{CORE} domain are off.	1. The V_{CORE} domain is power off.

Mode	Sleep	Deep-sleep	Standby
		2. Disable IRC16M, HXTAL and PLL.	2. Disable IRC16M, HXTAL and PLL.
LDO Status	On (normal power mode, normal driver mode)	On (normal power mode or low power mode, normal driver mode or low driver mode)	Off
Configuration	SLEEPDEEP = 0	SLEEPDEEP = 1 STBMOD = 0	SLEEPDEEP = 1 STBMOD = 1, WURST = 1
Entry	WFI or WFE	WFI or WFE	WFI or WFE
Wakeup	Any interrupt for WFI Any event (or interrupt when SEVONPEND is 1) for WFE	Any interrupt from EXTI lines for WFI. Any event(or interrupt when SEVONPEND is 1) from EXTI for WFE.	1. NRST pin 2. WKUP pin 3. FWDGT reset 4. RTC
Wakeup Latency	None	IRC16M wakeup time, LDO wakeup time added if LDO is in low power mode	Power on sequence

Note: In Standby mode, all I / Os are in high-impedance state except NRST pin, PC13 pin and PI8 when configured for RTC function, PC14 and PC15 pins when used as LXTAL crystal oscillator pins, and WKUP pin if enabled.

4.4. Register definition

PMU base address: 0x4000 7000

4.4.1. Control register (PMU_CTL)

Address offset: 0x00

Reset value: 0x0000 C000 (reset by wakeup from Standby 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
Reserved												LDEN[1:0]		HDS	HDEN
												rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDOVS[1:0]		Reserved		LDNP	LDLP	Reserved	BKPWEN	LVDT[2:0]			LVDEN	STBRST	WURST	STBMOD	LDOLP
rs				rw	rw		rw	rw			rw	rc_w1	rc_w1	rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:18	LDEN[1:0]	Enable low-driver mode in Deep-sleep mode 00: disable low-driver mode in Deep-sleep mode 01: Reserved 10: Reserved 11: enable low-driver mode in Deep-sleep mode
17	HDS	High-driver mode switch Set this bit by software only when HDRF flag is set and IRC16M or HXTAL used as system clock. After this bit is set, the system enters High-driver mode. This bit can be cleared by software. And cleared by hardware when exit from Deep-sleep mode or when the HDEN bit is clear. 0: No high-driver mode switch 1: High-driver mode switch
16	HDEN	High-driver mode enable This bit is set by software only when IRC16M or HXTAL used as system clock. This bit is cleared by software or by hardware when exit from Deep-sleep mode. 0: High-driver mode disable 1: High-driver mode enable
15:14	LDOVS[1:0]	LDO output voltage select These bits are set by software when the main PLL closed. 00: Reserved (LDO output voltage low mode(1.1V) 01: LDO output voltage low mode(1.1V) 10: LDO output voltage mid mode(1.2V)

		11: LDO output voltage high mode(1.2V)
13:12	Reserved	Must be kept at reset value.
11	LDNP	Low-driver mode when use normal power LDO 0: Normal-driver when use normal power LDO 1: Low-driver mode enabled when LDEN is 11 and use normal power LDO
10	LDLP	Low-driver mode when use low power LDO. 0: Normal-driver when use low power LDO 1: Low-driver mode enabled when LDEN is 11 and use low power LDO
9	Reserved	Must be kept at reset value.
8	BKPWEN	Backup domain write enable 0: Disable write access to the registers in Backup domain 1: Enable write access to the registers in Backup domain 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 000: 2.1V 001: 2.3V 010: 2.4V 011: 2.6V 100: 2.7V 101: 2.9V 110: 3.0V 111: 3.1V
4	LVDEN	Low voltage detector enable 0: Disable low voltage detector 1: Enable low voltage detector
3	STBRST	Standby flag reset 0: No effect 1: Reset the standby flag This bit is always read as 0.
2	WURST	Wakeup flag reset 0: No effect 1: Reset the wakeup flag This bit is always read as 0.
1	STBMOD	Standby mode 0: Enter the Deep-sleep mode when the Cortex®-M33 enters SLEEPDEEP mode 1: Enter the Standby mode when the Cortex®-M33 enters SLEEPDEEP mode
0	LDOLP	LDO low power Mode 0: The LDO operates normally during the Deep-sleep mode

1: The LDO is in low power mode during the Deep-sleep mode

4.4.2. Control and status register (PMU_CS)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby 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
Reserved												LDRF[1:0]		HDSRF	HDRF
												rc_w1		r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LDOVSRF	Reserved				BLDOON	WUPEN	Reserved				BLDORF	LVDF	STBF	WUF
	r					rw	rw					r	r	r	r

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:18	LDRF[1:0]	Low-driver mode ready flag These bits are set by hardware when enter Deep-sleep mode and the LDO in Low-driver mode. These bits are cleared by software when write 11. 00: normal driver in Deep-sleep mode 01: Reserved 10: Reserved 11: Low-driver mode in Deep-sleep mode
17	HDSRF	High-driver switch ready flag 0: High-driver switch not ready 1: High-driver switch ready
16	HDRF	High-driver ready flag 0: High-driver not ready 1: High-driver ready
15	Reserved	Must be kept at reset value
14	LDOVSRF	LDO voltage select ready flag 0: LDO voltage select not ready 1: LDO voltage select ready
13:10	Reserved	Must be kept at reset value.
9	BLDOON	Backup SRAM LDO on This bit is set by software to open Backup SRAM LDO for data protection of backup SRAM when V_{DD} shut down. When V_{DD} shut down and this bit is cleared, the data in Backup SRAM will be lost. 0: Backup SRAM LDO closed

		1: Open the Backup SRAM LDO
8	WUPEN	WKUP Pin (PA0) enable 0: Disable WKUP pin (PA0) function 1: Enable WKUP pin (PA0) function If WUPEN is set before entering the Standby mode, a rising edge on the WKUP pin wakes up the system from the Standby mode. As the WKUP pin is active high, the WKUP pin is internally configured to input pull down mode. And set this bit will trigger a wakup event when the input changes to high.
7:4	Reserved	Must be kept at reset value.
3	BLDORF	Backup SRAM LDO ready flag 0: Backup SRAM LDO not ready 1: Backup SRAM LDO ready
2	LVDF	Low voltage detector status flag 0: Low Voltage event has not occurred (V_{DD} is higher than the specified LVD threshold) 1: Low Voltage event occurred (V_{DD} is equal to or lower than the specified LVD threshold) **Note:** The LVD function is stopped in standby mode.
1	STBF	Standby flag 0: The device has not entered the Standby mode 1: The device has been in the Standby mode This bit is cleared only by a POR / PDR or by setting the STBRST bit in the PMU_CTL register.
0	WUF	Wakeup flag 0: No wakeup event has been received 1: Wakeup event occurred from the WKUP pin or the RTC wakeup event including RTC Tamper event, RTC alarm event, RTC Time Stamp event or RTC Wakeup. This bit is cleared only by a POR / PDR or by setting the WURST bit in the PMU_CTL register.

5. Reset and clock unit (RCU)

5.1. Reset control unit (RCTL)

5.1.1. Overview

GD32F527xx reset control 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.

5.1.2. 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 1.2V 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:

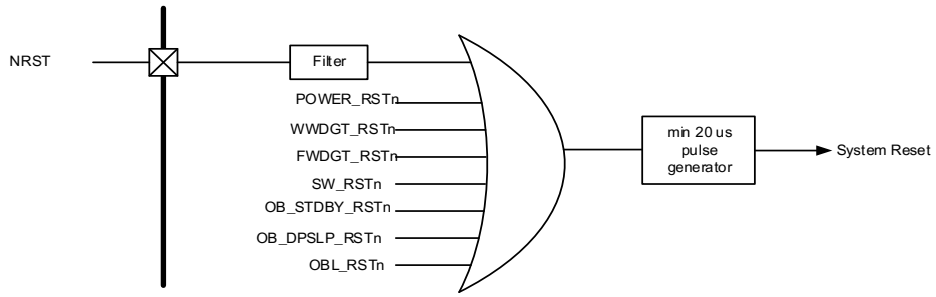
- A power reset (POWER_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT_RSTn).
- A free watchdog timer reset (FWDGT_RSTn).
- The SYSRESETREQ bit in Cortex®-M33 application interrupt and reset control register is set (SW_RSTn).
- Reset generated when entering Standby mode when 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 5-1. 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 (V_{DD} or V_{BAT} power on, if both supplies have previously been powered off).

Note: The BKPSRAM is not reset by backup domain reset.

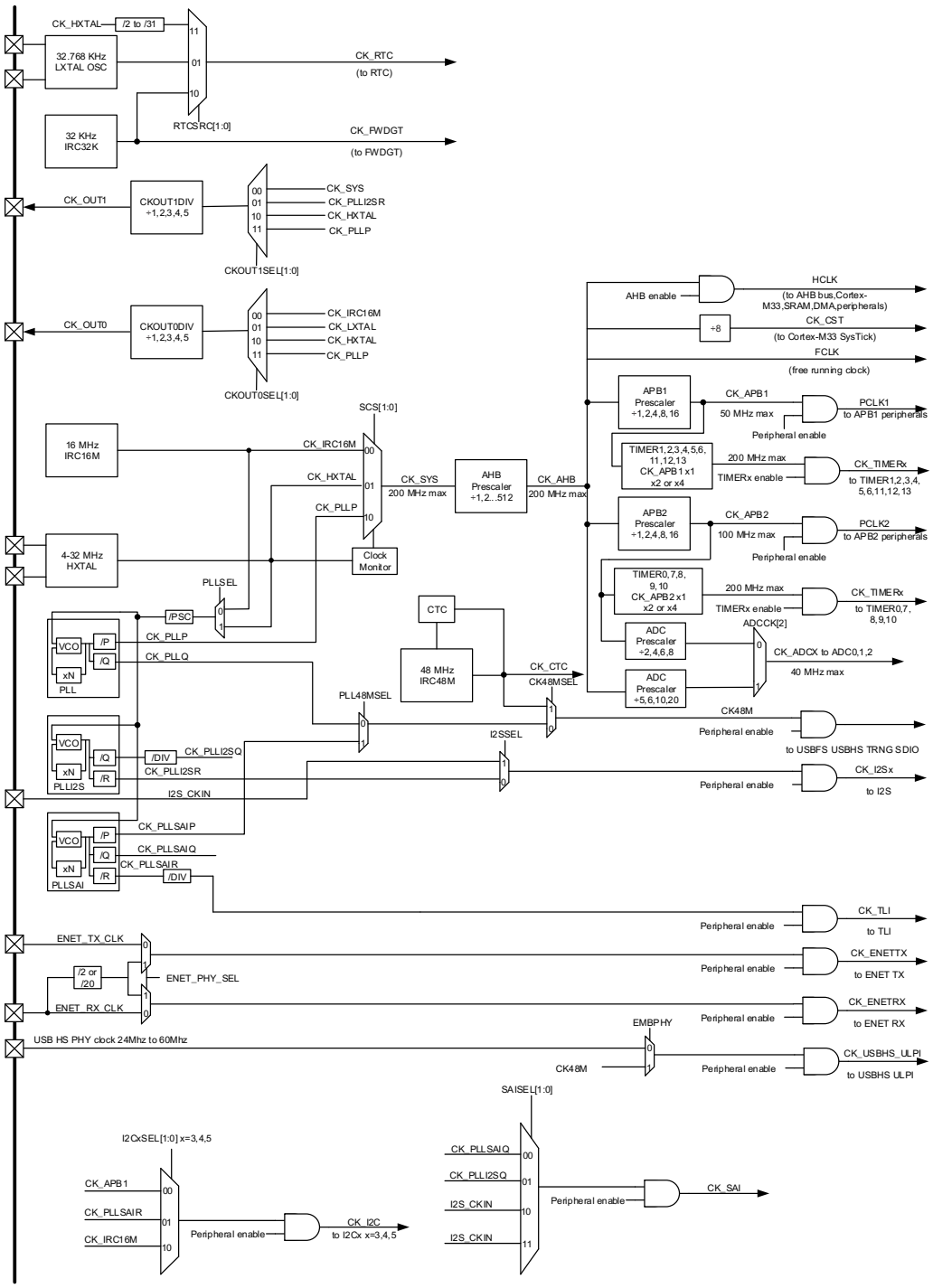
5.2. Clock control unit (CCTL)

5.2.1. Overview

The clock control unit provides a range of frequencies and clock functions. These include a Internal 16M RC oscillator (IRC16M), 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), three Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex®-M33 are derived from the system clock (CK_SYS) which can source from the IRC16M, HXTAL or PLL. The maximum operating frequency of the system clock (CK_SYS) can be up to 200 MHz.

Figure 5-2. Clock tree



The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB and the APB2/APB1 domains is 200 MHz/100 MHz/50 MHz. The Cortex System Timer (SysTick) external clock is clocked with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the AHB clock (HCLK), configurable in the systick control and status register.

The ADCs are clocked by the clock of APB2 divided by 2, 4, 6, 8 or by the clock of AHB

divided by 5, 6, 10, 20, which defined by ADCCK in ADC_SYNCCTL register.

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 USBFS/USBHS/TRNG/SDIO are clocked by the clock of CK48M. The CK48M is selected from the clock of PLLQ, the clock of PLLSAIP or the clock of IRC48M by PLL48MSEL and CK48MSEL bit in RCU_ADDCTL register.

The USBHS ULPI is clocked by external ULPI PHY clock or CK48M, which select by EMBPHY in USBHS_ GUSBCS register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

The I2S is clocked by the clock of PLLI2SR or External PIN I2S_CKIN which defined by I2SSEL bit in RCU_CFG0 register.

The I2C is clocked by the clock of CK_APB1, CK_PLLSAIR, CK_IRC16M which defined by I2CxSEL bits in RCU_CFG2 register.

The TLI is clocked by the clock of PLLSAIR divided by 2, 4, 8, 16 which defined by PLLSAIRDIV bits in RCU_CFG1 register.

The ENET TX/RX are clocked by External PIN (ENET_TX_CLK / ENET_RX_CLK), which select by ENET_PHY_SEL bit in SYSCFG_CFG1 register.

The RTC is clocked by LXTAL clock or IRC32K clock or HXTAL clock divided by 2 to 31 (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 31 (defined by RTCDIV bits in RCU_CFG0), the clock disappeared when the 1.2V core domain power off. After the RTC select IRC32K, the clock disappeared when VDD power off. When the RTC select LXTAL, the clock disappeared when VDD and VBAT power off.

The FWDGT is clocked by IRC32K clock, which is forced on when FWDGT started.

5.2.2. Characteristics

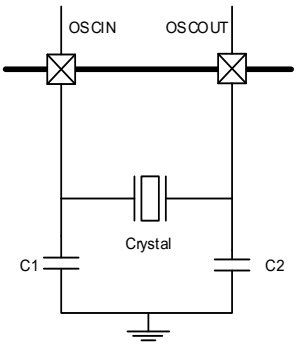
- 4 to 32 MHz High Speed crystal oscillator (HXTAL).
- Internal 16 MHz RC oscillator (IRC16M).
- Internal 48 MHz RC oscillator (IRC48M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 32KHz RC oscillator (IRC32K).
- PLL clock source can be HXTAL or IRC16M.
- HXTAL clock monitor.

5.2.3. Function overview

High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), which has a frequency from 4 to 32 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.

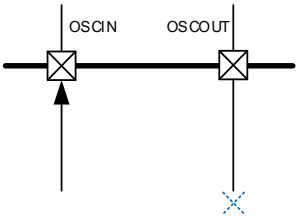
Figure 5-3. HXTAL clock source



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. During bypass mode, the signal is connected to OSCIN, and OSCOUT remains in the suspended state, as shown in ***Figure 5-4. HXTAL clock source in bypass mode***. The CK_HXTAL is equal to the external clock which drives the OSCIN pin.

Figure 5-4. HXTAL clock source in bypass mode



Internal 16M RC oscillators (IRC16M)

The internal 16M RC oscillator, IRC16M, has a fixed frequency of 16 MHz and is the default

clock source selection for the CPU when the device is powered up. The IRC16M oscillator provides a lower cost type clock source as no external components are required. The IRC16M RC oscillator can be switched on or off using the IRC16MEN bit in the control register RCU_CTL. The IRC16MSTB flag in the control register RCU_CTL is used to indicate if the internal 16M RC oscillator is stable. The start-up time of the IRC16M oscillator is shorter than the HXTAL crystal oscillator. An interrupt can be generated if the related interrupt enable bit, IRC16MSTBIE, in the clock interrupt register, RCU_INT, is set when the IRC16M becomes stable. The IRC16M clock can also be used as the system clock source or the PLL input clock.

The frequency accuracy of the IRC16M 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 PLL is the system clock source, to minimize the time required for the system to recover from the Deep-sleep Mode, the hardware forces the IRC16M clock to be the system 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 USBFS/USBHS/TRNG/SDIO used. The IRC48M RC oscillator can be switched on or off using the IRC48MEN bit in the RCU_ADDCTL register. The IRC48MSTB flag in the RCU_ADDCTL 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 USBFS/USBHS/TRNG/SDIO.

The frequency accuracy of the IRC48M can be calibrated by the manufacturer, but its operating frequency is still not enough accurate because the USB need the frequency must between 48MHz+/−1%. A hardware automatically dynamic trim performed in CTC unit adjust the IRC48M to the needed frequency.

Phase locked loop (PLL)

There are three internal Phase Locked Loop, the PLL, PLLI2S and PLLSAI. The PLLP could be used to generator system clock (no more than 200MHz) and PLLQ clock which used to USBFS/USBHS/TRNG/SDIO. The PLLI2S is used to generator the clock to I2S. The PLLSAI is used to generator the clock to CK48M or TLI.

The PLL can be switched on or off by using the PLLEN bit in the RCU_CTL register. The PLLSTB flag in the RCU_CTL register will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSTBIE, in the RCU_INT register, is set as the PLL becomes stable.

The PLLI2S can be switched on or off by using the PLLI2SEN bit in the RCU_CTL register. The PLLI2SSTB flag in the RCU_CTL register will indicate if the PLLI2S clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLI2SSTBIE, in the RCU_INT

register, is set as the PLLI2S becomes stable.

The PLLSAI can be switched on or off by using the PLLSAIEN bit in the RCU_CTL register. The PLLSAISTB flag in the RCU_CTL register will indicate if the PLLSAI clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSAISTBIE, in the RCU_INT register, is set as the PLLSAI becomes stable.

The three 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.

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 free 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.

System clock (CK_SYS) selection

After the system reset, the default CK_SYS source will be IRC16M and can be switched to HXTAL or CK_PLLP 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 PLL) 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®-M33. If the HXTAL is selected as the clock source of CK_SYS or PLL and CK_PLLP used as system clock, the HXTAL failure will force the CK_SYS source to IRC16M and the PLL 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.

Note: After CKM is enabled, when a HXTAL fault is detected, the NMI interrupt of the system will be triggered, and the enable bits and stability flag bits of HXTAL and PLL will be reset to zero. At this time, the system clock will automatically switch to IRC16M, and the enable bits of HXTAL and PLL cannot be restored.

Clock output capability

The clock output capability is ranging from 32 kHz to 200 MHz. There are several clock signals can be selected via the CK_OUT0 clock source selection bits, CKOUT0SEL, in the clock configuration register 0 (RCU_CFG0). 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 0 (RCU_CFG0).

Note: The clock output function has a deviation in the output clock and cannot be used as a high-precision clock source.

Table 5-1. Clock output 0 source select

Clock Source 0 Selection bits	Clock Source
00	CK_IRC16M
01	CK_LXTAL
10	CK_HXTAL
11	CK_PLLP

Table 5-2. Clock output 1 source select

Clock Source 1 Selection bits	Clock Source
00	CK_SYS
01	CK_PLLI2SR
10	CK_HXTAL
11	CK_PLLP

The CK_OUT0 frequency can be reduced by a configurable binary divider, controlled by the CKOUT0DIV bits, in the clock configuration register (RCU_CFG0).

The CK_OUT1 frequency can be reduced by a configurable binary divider, controlled by the CKOUT1DIV bits, in the clock configuration register (RCU_CFG0).

RTC clock measure

The three clock source of RTC clock, LXTAL, IRC32K, HXTAL divided by 2 to 31 (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 CI3_RMP in ***TIMER4_IRMP*** register and ITI1_RMP in ***TIMER10_IRMP*** register for detail.

Voltage control

The 1.2V domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU_DSV).

Table 5-3. 1.2V domain voltage selected in deep-sleep mode

DSLPVS[2:0]	Deep-sleep mode voltage(V)
000	default value
001	default value-0.1
010	default value-0.2
011	default value-0.3
100 ~ 111	reserved

The RCU_DSV register are protected by voltage key register (RCU_VKEY). Only after write 0x1A2B3C4D to the RCU_VKEY, the RCU_DSV register can be written.

5.3. Register definition

RCU base address: 0x4002 3800

5.3.1. Control register (RCU_CTL)

Address offset: 0x00

Reset value: 0x0000 xx83 where x is 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
Reserved		PLLSAISTB	PLLSAIEN	PLLI2SSTB	PLLI2SEN	PLLSTB	PLLEN	Reserved				CKMEN	HXTALBPS	HXTALSTB	HXTALEN
		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
IRC16MCALIB[7:0]								IRC16MADJ[4:0]					Reserved	IRC16MSTB	IRC16MEN
r								rw						r	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	PLLSAISTB	PLLSAI clock stabilization flag Set by hardware to indicate if the PLLSAI output clock is stable and ready for use. 0: PLLSAI is not stable 1: PLLSAI is stable
28	PLLSAIEN	PLLSAI enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLLSAI is switched off 1: PLLSAI is switched on
27	PLLI2SSTB	PLLI2S clock stabilization flag Set by hardware to indicate if the PLLI2S output clock is stable and ready for use. 0: PLLI2S is not stable 1: PLLI2S is stable
26	PLLI2SEN	PLLI2S enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLLI2S is switched off 1: PLLI2S is switched on
25	PLLSTB	PLL clock stabilization flag Set by hardware to indicate if the PLL output clock is stable and ready for use.

		0: PLL is not stable 1: PLL is stable
24	PLLEN	PLL enable Set and reset by software. This bit cannot be reset if the PLL clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL is switched off 1: PLL is switched on
23:20	Reserved	Must be kept at reset value.
19	CKMEN	HXTAL clock monitor enable 0: Disable the High speed 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the High speed 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor 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 IRC16M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software. **Note**: When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC16M internal RC oscillator regardless of the control bit, IRC16MEN, state.
18	HXTALBPS	High speed crystal oscillator (HXTAL) clock bypass mode enable The HXTALBPS bit can be written only if the HXTALEN is 0. 0: Disable the HXTAL Bypass mode 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 Set by hardware to indicate if the HXTAL oscillator is stable and ready for use. 0: HXTAL oscillator is not stable 1: HXTAL oscillator is stable
16	HXTALEN	High Speed crystal oscillator (HXTAL) enable Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL input clock when PLL clock is selected to the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: High speed 4 ~ 32 MHz crystal oscillator disabled 1: High speed 4 ~ 32 MHz crystal oscillator enabled
15:8	IRC16MCALIB[7:0]	Internal 16MHz RC Oscillator calibration value register These bits are load automatically at power on.
7:3	IRC16MADJ[4:0]	Internal 16MHz RC Oscillator clock trim adjust value These bits are set by software. The trimming value is these bits (IRC16MADJ) added to the IRC16MCALIB[7:0] bits. The trimming value should trim the IRC16M to 16 MHz +/- 1%.

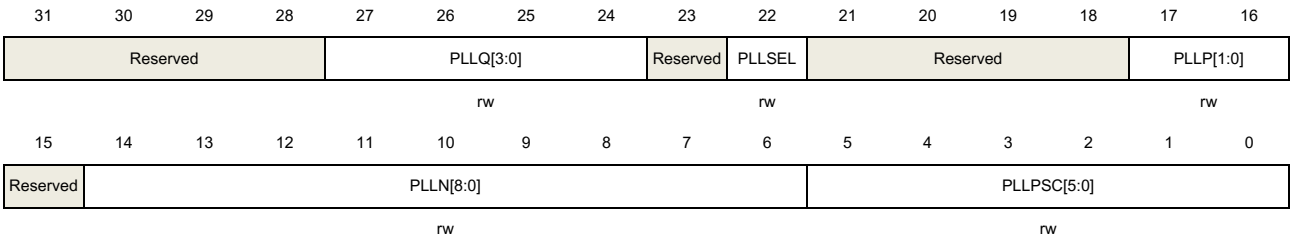
2	Reserved	Must be kept at reset value.
1	IRC16MSTB	IRC16M Internal 16MHz RC Oscillator stabilization flag Set by hardware to indicate if the IRC16M oscillator is stable and ready for use. 0: IRC16M oscillator is not stable 1: IRC16M oscillator is stable
0	IRC16MEN	Internal 16MHz RC oscillator enable Set and reset by software. This bit cannot be reset if the IRC16M 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. 0: Internal 16 MHz RC oscillator disabled 1: Internal 16 MHz RC oscillator enabled

5.3.2. PLL register (RCU_PLL)

Address offset: 0x04
Reset value: 0x2400 3010

To configure the PLL clock, refer to the following formula:
CK_PLLVCOSRC = CK_PLLSRC / PLLPSC
CK_PLLVCO = CK_PLLVCOSRC × PLLN
CK_PLLP = CK_PLLVCO / PLLP
CK_PLLQ = CK_PLLVCO / PLLQ

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	PLLQ[3:0]	The PLL Q output frequency division factor from PLL VCO clock Set and reset by software when the PLL is disable. These bits used to generator PLL Q output clock (CK_PLLQ) from PLL VCO clock (CK_PLLVCO). The CK_PLLQ is used to UBSFS/USBHS (48MHz), TRNG (48MHz), or SDIO (<=48MHz). The CK_PLLVCO is described in PLLN bits in RCU_PLL register. 0000: Reserved 0001: Reserved 0010: CK_PLLQ = CK_PLLVCO / 2 0011: CK_PLLQ = CK_PLLVCO / 3 0100: CK_PLLQ = CK_PLLVCO / 4

…
1111: CK_PLLQ = CK_PLLVCO / 15

23	Reserved	Must be kept at reset value.
22	PLLSEL	PLL clock source selection Set and reset by software to control the PLL clock source. 0: IRC16M clock selected as source clock of PLL, PLLSAI, PLLI2S 1: HXTAL clock selected as source clock of PLL, PLLSAI, PLLI2S
21:18	Reserved	Must be kept at reset value.
17:16	PLLP[1:0]	The PLLP output frequency division factor from PLL VCO clock Set and reset by software when the PLL is disable. These bits used to generator PLLP output clock (CK_PLLP) from PLL VCO clock (CK_PLLVCO). The CK_PLLP is used to system clock (no more than 200MHz). The CK_PLLVCO is described in PLLN bits in RCU_PLL register. 00 : CK_PLLP = CK_PLLVCO / 2 01 : CK_PLLP = CK_PLLVCO / 4 10 : CK_PLLP = CK_PLLVCO / 6 11 : CK_PLLP = CK_PLLVCO / 8
15	Reserved	Must be kept at reset value.
14:6	PLLN[8:0]	The PLL VCO clock multiplication factor Set and reset by software (only use word/half-word write) when the PLL is disable. These bits used to generator PLL VCO clock (CK_PLLVCO) from PLL VCO source clock (CK_PLLVCOSRC). The CK_PLLVCOSRC is described in PLLPSC bits in RCU_PLL register. Note: The frequency of CK_PLLVCO is between 100MHz to 500MHz The value of PLLN must : 64<=PLLN<=500 (when SSCGON=0 in RCU_PLLSSCTL) 69<=PLLN<=500 (when SSCGON=1/SS_TYPE=0 in RCU_PLLSSCTL) 71<=PLLN<=500 (when SSCGON=1/SS_TYPE=1 in RCU_PLLSSCTL) 000000000: Reserved 000000001: Reserved … 000111111: Reserved 001000000: CK_PLLVCO = CK_PLLVCOSRC x 64. 001000001: CK_PLLVCO = CK_PLLVCOSRC x 65. … 111110100: CK_PLLVCO = CK_PLLVCOSRC x 500. 111110101: Reserved … 111111111: Reserved
5:0	PLLPSC[5:0]	The PLL VCO source clock prescaler Set and reset by software when the PLL is disable. These bits used to generate the

clock of PLL VCO source clock (CK_PLLVCOSRC), PLLSAI VCO source clock (CK_PLLSAIVCOSRC), or PLLI2S VCO source clock (CK_PLLI2SVCOSRC) from PLL source clock (CK_PLLSRC) which described in PLLSEL in RCU_PLL register. The VCO source clock is between 1M to 2MHz.

000000: Reserved.

000001: Reserved

000010: CK_PLLSRC / 2

000011: CK_PLLSRC / 3

…

111111: CK_PLLSRC / 63

5.3.3. 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
CKOUT1SEL[1:0]	CKOUT1DIV[2:0]	CKOUT0DIV[2:0]	I2SSEL	CKOUT0SEL[1:0]	RTCDIV[4:0]
rw	rw	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	CKOUT1SEL[1:0]	CKOUT1 clock source selection Set and reset by software. 00: System clock selected 01: CK_PLLI2SR clock selected 10: High Speed crystal oscillator clock (HXTAL) selected 11: CK_PLLP clock selected
29:27	CKOUT1DIV[2:0]	The CK_OUT1 divider which the CK_OUT1 frequency can be reduced see bits 31:30 of RCU_CFG0 for CK_OUT1 0xx: The CK_OUT1 is divided by 1 100: The CK_OUT1 is divided by 2 101: The CK_OUT1 is divided by 3 110: The CK_OUT1 is divided by 4 111: The CK_OUT1 is divided by 5
26:24	CKOUT0DIV[2:0]	The CK_OUT0 divider which the CK_OUT0 frequency can be reduced see bits 22:21 of RCU_CFG0 for CK_OUT0 0xx: The CK_OUT0 is divided by 1 100: The CK_OUT0 is divided by 2

		101: The CK_OUT0 is divided by 3 110: The CK_OUT0 is divided by 4 111: The CK_OUT0 is divided by 5
23	I2SSEL	I2S clock source selection Set and reset by software to control the I2S clock source. 0: CK_PLLI2SR clock selected as I2S source clock 1: External I2S_CKIN PIN selected as I2S source clock
22:21	CKOUT0SEL[1:0]	CKOUT0 clock source selection Set and reset by software. 00: Internal 16M RC oscillator clock selected 01: Low speed crystal oscillator clock (LXTAL) selected 10: High speed crystal oscillator clock (HXTAL) selected 11: CK_PLLP clock selected
20:16	RTCDIV[4:0]	RTC clock divider factor Set and reset by software. These bits is used to generator clock for RTC (no more than 1MHz) from HXTAL clock. 00000: no clock for RTC 00001: no clock for RTC 00010: CK_HXTAL / 2 00011: CK_HXTAL / 3 … 11111: CK_HXTAL / 31
15:13	APB2PSC[2:0]	APB2 prescaler selection Set and reset by software to control the APB2 clock division ratio. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
12:10	APB1PSC[2:0]	APB1 prescaler selection Set and reset by software to control the APB1 clock division ratio. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
9:8	Reserved	Must be kept at reset value.
7:4	AHBPSC[3:0]	AHB prescaler selection Set and reset by software to control the AHB clock division ratio 0xxx: CK_SYS selected 1000: (CK_SYS / 2) selected

		1001: (CK_SYS / 4) selected 1010: (CK_SYS / 8) selected 1011: (CK_SYS / 16) selected 1100: (CK_SYS / 64) selected 1101: (CK_SYS / 128) selected 1110: (CK_SYS / 256) selected 1111: (CK_SYS / 512) selected
3:2	SCSS[1:0]	System clock switch status Set and reset by hardware to indicate the clock source of system clock. 00: select CK_IRC16M as the CK_SYS source 01: select CK_HXTAL as the CK_SYS source 10: select CK_PLLP as the CK_SYS source 11: reserved
1:0	SCS[1:0]	System clock switch 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 IRC16M 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_IRC16M as the CK_SYS source 01: select CK_HXTAL as the CK_SYS source 10: select CK_PLLP as the CK_SYS source 11: reserved

5.3.4. 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								CKMIC	PLLSAI STBIC	PLLI2S STBIC	PLL STBIC	HXTAL STBIC	IRC16M STBIC	LXTAL STBIC	IRC32K STBIC
								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	PLLSAI STBIE	PLLI2S STBIE	PLL STBIE	HXTAL STBIE	IRC16M STBIE	LXTAL STBIE	IRC32K STBIE	CKMIF	PLLSAI STBIF	PLLI2S STBIF	PLL STBIF	HXTAL STBIF	IRC16M STBIF	LXTAL STBIF	IRC32K STBIF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	CKMIC	HXTAL clock stuck interrupt clear

		Write 1 by software to reset the CKMIF flag. 0: Not reset CKMIF flag 1: Reset CKMIF flag
22	PLLSAISTBIC	PLLSAI stabilization interrupt clear Write 1 by software to reset the PLLSAISTBIF flag. 0: Not reset PLLSAISTBIF flag 1: Reset PLLSAISTBIF flag
21	PLLI2SSTBIC	PLLI2S stabilization interrupt clear Write 1 by software to reset the PLLI2SSTBIF flag. 0: Not reset PLLI2SSTBIF flag 1: Reset PLLI2SSTBIF flag
20	PLLSTBIC	PLL stabilization interrupt clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL stabilization interrupt clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC16MSTBIC	IRC16M stabilization interrupt clear Write 1 by software to reset the IRC16MSTBIF flag. 0: Not reset IRC16MSTBIF flag 1: Reset IRC16MSTBIF flag
17	LXTALSTBIC	LXTAL stabilization interrupt clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALSTBIF flag
16	IRC32KSTBIC	IRC32K stabilization interrupt clear Write 1 by software to reset the IRC32KSTBIF flag. 0: Not reset IRC32KSTBIF flag 1: Reset IRC32KSTBIF flag
15	Reserved	Must be kept at reset value.
14	PLLSAISTBIE	PLLSAI stabilization interrupt enable Set and reset by software to enable/disable the PLLSAI stabilization interrupt. 0: Disable the PLLSAI stabilization interrupt 1: Enable the PLLSAI stabilization interrupt
13	PLLI2SSTBIE	PLLI2S stabilization interrupt enable Set and reset by software to enable/disable the PLLI2S stabilization interrupt. 0: Disable the PLLI2S stabilization interrupt

		1: Enable the PLLI2S stabilization interrupt
12	PLLSTBIE	PLL stabilization interrupt enable Set and reset by software to enable/disable the PLL stabilization interrupt. 0: Disable the PLL stabilization interrupt 1: Enable the PLL stabilization interrupt
11	HXTALSTBIE	HXTAL stabilization interrupt enable Set and reset by software to enable/disable the HXTAL stabilization interrupt 0: Disable the HXTAL stabilization interrupt 1: Enable the HXTAL stabilization interrupt
10	IRC16MSTBIE	IRC16M stabilization interrupt enable Set and reset by software to enable/disable the IRC16M stabilization interrupt 0: Disable the IRC16M stabilization interrupt 1: Enable the IRC16M stabilization interrupt
9	LXTALSTBIE	LXTAL stabilization interrupt enable LXTAL stabilization interrupt enable/disable control 0: Disable the LXTAL stabilization interrupt 1: Enable the LXTAL stabilization interrupt
8	IRC32KSTBIE	IRC32K stabilization interrupt enable IRC32K stabilization interrupt enable/disable control 0: Disable the IRC32K stabilization interrupt 1: Enable the IRC32K stabilization interrupt
7	CKMIF	HXTAL clock stuck interrupt flag Set by hardware when the HXTAL clock is stuck. Reset when setting the CKMIC bit by software. 0: Clock operating normally 1: HXTAL clock stuck
6	PLLSAISTBIF	PLLSAI stabilization interrupt flag Set by hardware when the PLLSAI is stable and the PLLSAISTBIE bit is set. Reset when setting the PLLSAISTBIC bit by software. 0: No PLLSAI stabilization interrupt generated 1: PLLSAI stabilization interrupt generated
5	PLLI2SSTBIF	PLLI2S stabilization interrupt flag Set by hardware when the PLLI2S is stable and the PLLI2SSTBIE bit is set. Reset when setting the PLLI2SSTBIC bit by software. 0: No PLLI2S stabilization interrupt generated 1: PLLI2S stabilization interrupt generated
4	PLLSTBIF	PLL stabilization interrupt flag Set by hardware when the PLL is stable and the PLLSTBIE bit is set. Reset when setting the PLLSTBIC bit by software. 0: No PLL stabilization interrupt generated

		1: PLL stabilization interrupt generated
3	HXTALSTBIF	HXTAL stabilization interrupt flag Set by hardware when the High speed 4 ~ 32 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set. Reset when setting the HXTALSTBIC bit by software. 0: No HXTAL stabilization interrupt generated 1: HXTAL stabilization interrupt generated
2	IRC16MSTBIF	IRC16M stabilization interrupt flag Set by hardware when the Internal 16 MHz RC oscillator clock is stable and the IRC16MSTBIE bit is set. Reset when setting the IRC16MSTBIC bit by software. 0: No IRC16M stabilization interrupt generated 1: IRC16M stabilization interrupt generated
1	LXTALSTBIF	LXTAL stabilization interrupt flag Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set. Reset when setting the LXTALSTBIC bit by software. 0: No LXTAL stabilization interrupt generated 1: LXTAL stabilization interrupt generated
0	IRC32KSTBIF	IRC32K stabilization interrupt flag Set by hardware when the Internal 32kHz RC oscillator clock is stable and the IRC32KSTBIE bit is set. Reset when setting the IRC32KSTBIC bit by software. 0: No IRC32K stabilization clock ready interrupt generated 1: IRC32K stabilization interrupt generated

5.3.5. 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		USBHSRST	Reserved			ENETRST	Reserved	IPARST	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			CRCRST	Reserved			PIRST	PHRST	PGRST	PFRST	PERST	PDRST	PCRST	PBRST	PARST
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.

29	USBHSRST	USBHS reset This bit is set and reset by software. 0: No reset 1: Reset the USBHS
28:26	Reserved	Must be kept at reset value.
25	ENETRST	Ethernet reset This bit is set and reset by software. 0: No reset 1: Reset the Ethernet
24	Reserved	Must be kept at reset value.
23	IPARST	IPA reset This bit is set and reset by software. 0: No reset 1: Reset the IPA
22	DMA1RST	DMA1 reset This bit is set and reset by software. 0: No reset 1: Reset the DMA1
21	DMA0RST	DMA0 reset This bit is set and reset by software. 0: No reset 1: Reset the DMA0
20:13	Reserved	Must be kept at reset value.
12	CRCRST	CRC reset This bit is set and reset by software. 0: No reset 1: Reset the CRC
11:9	Reserved	Must be kept at reset value.
8	PIRST	GPIO port I reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port I
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 This bit is set and reset by software.

		0: No reset 1: Reset the GPIO port G
5	PFRST	GPIO port F reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port F
4	PERST	GPIO port E reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port E
3	PDRST	GPIO port D reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port D
2	PCRST	GPIO port C reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port C
1	PBRST	GPIO port B reset This bit is set and reset by software. 0: No reset 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

5.3.6. 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								USBFSRST	TRNGRST	HAURST	CAURST	PKCAURST	Reserved		DCIRST
								rw	rw	rw	rw	rw			rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	USBFSRST	USBFS reset This bit is set and reset by software. 0: No reset 1: Reset the USBFS
6	TRNGRST	TRNG reset This bit is set and reset by software. 0: No reset 1: Reset the TRNG
5	HAURST	HAU reset This bit is set and reset by software. 0: No reset 1: Reset the HAU
4	CAURST	CAU reset This bit is set and reset by software. 0: No reset 1: Reset the CAU
3	PKCAURST	PKCAU reset This bit is set and reset by software. 0: No reset 1: Reset the PKCAU
2:1	Reserved	Must be kept at reset value.
0	DCIRST	DCI reset This bit is set and reset by software. 0: No reset 1: Reset the DCI

5.3.7. 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															EXMCRST
															rw

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	EXMCRST	EXMC reset This bit is set and reset by software. 0: No reset 1: Reset the EXMC

5.3.8. 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	PMURST	Reserved	CAN1RST	CAN0RST	Reserved	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	I2C5RST	I2C4RST	WWDGTRST	I2C3RST	Reserved	TIMER13RST	TIMER12RST	TIMER11RST	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 This bit is set and reset by software. 0: No reset 1: Reset the UART7
30	UART6RST	UART6 reset This bit is set and reset by software. 0: No reset 1: Reset the UART6
29	DACRST	DAC reset This bit is set and reset by software. 0: No reset 1: Reset the DAC
28	PMURST	PMU reset This bit is set and reset by software. 0: No reset 1: Reset the PMU
27	Reserved	Must be kept at reset value.

26	CAN1RST	CAN1 reset This bit is set and reset by software. 0: No reset 1: Reset the CAN1
25	CAN0RST	CAN0 reset This bit is set and reset by software. 0: No reset 1: Reset the CAN0
24	Reserved	Must be kept at reset value.
23	I2C2RST	I2C2 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C2
22	I2C1RST	I2C1 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C1
21	I2C0RST	I2C0 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C0
20	UART4RST	UART4 reset This bit is set and reset by software. 0: No reset 1: Reset the UART4
19	UART3RST	UART3 reset This bit is set and reset by software. 0: No reset 1: Reset the UART3
18	USART2RST	USART2 reset This bit is set and reset by software. 0: No reset 1: Reset the USART2
17	USART1RST	USART1 reset This bit is set and reset by software. 0: No reset 1: Reset the USART1
16	Reserved	Must be kept at reset value.
15	SPI2RST	SPI2 reset

This bit is set and reset by software.
0: No reset
1: Reset the SPI2

14	SPI1RST	SPI1 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI1
13	I2C5RST	I2C5 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C5
12	I2C4RST	I2C4 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C4
11	WWDGTRST	WWDGT reset This bit is set and reset by software. 0: No reset 1: Reset the WWDGT
10	I2C3RST	I2C3 reset This bit is set and reset by software. 0: No reset 1: Reset the I2C3
9	Reserved	Must be kept at reset value.
8	TIMER13RST	TIMER13 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER13
7	TIMER12RST	TIMER12 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER12
6	TIMER11RST	TIMER11 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER11
5	TIMER6RST	TIMER6 reset This bit is set and reset by software. 0: No reset

		1: Reset the TIMER6
4	TIMER5RST	TIMER5 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER5
3	TIMER4RST	TIMER4 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER4
2	TIMER3RST	TIMER3 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER3
1	TIMER2RST	TIMER2 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER2
0	TIMER1RST	TIMER1 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER1

5.3.9. 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
Reserved					TLIRST	Reserved			SAIRST	SPI5RST	SPI4RST	Reserved	TIMER10RST	TIMER9RST	TIMER8RST
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SYSCFGRST	SPI3RST	SPI0RST	SDIORST	Reserved		ADCRST	Reserved		USART5RST	USART0RST	Reserved		TIMER7RST	TIMER0RST
	rw	rw	rw	rw			rw			rw	rw			rw	rw

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	TLIRST	TLI reset This bit is set and reset by software.

		0: No reset 1: Reset the TLI
25:23	Reserved	Must be kept at reset value.
22	SAIRST	SAI reset This bit is set and reset by software. 0: No reset 1: Reset the SAI
21	SPI5RST	SPI5 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI5
20	SPI4RST	SPI4 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI4
19	Reserved	Must be kept at reset value.
18	TIMER10RST	TIMER10 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER10
17	TIMER9RST	TIMER9 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER9
16	TIMER8RST	TIMER8 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER8
15	Reserved	Must be kept at reset value.
14	SYSCFGRST	SYSCFG reset This bit is set and reset by software. 0: No reset 1: Reset the SYSCFG
13	SPI3RST	SPI3 reset This bit is set and reset by software. 0: No reset 1: Reset the SPI3
12	SPI0RST	SPI0 reset

		This bit is set and reset by software. 0: No reset 1: Reset the SPI0
11	SDIORST	SDIO reset This bit is set and reset by software. 0: No reset 1: Reset the SDIO
10:9	Reserved	Must be kept at reset value.
8	ADCRST	ADC reset This bit is set and reset by software. 0: No reset 1: Reset the all ADCs
7:6	Reserved	Must be kept at reset value.
5	USART5RST	USART5 reset This bit is set and reset by software. 0: No reset 1: Reset the USART5
4	USART0RST	USART0 reset This bit is set and reset by software. 0: No reset 1: Reset the USART0
3:2	Reserved	Must be kept at reset value.
1	TIMER7RST	TIMER7 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER7
0	TIMER0RST	TIMER0 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER0

5.3.10. AHB1 enable register (RCU_AHB1EN)

Address offset: 0x30

Reset value: 0x0010 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	USBHSULPIEN	USBHSEN	ENETPTPEN	ENETRXEN	ENETTXEN	ENETEN	Reserved	IPAEN	DMA1EN	DMA0EN	TCMSRAMEN	Reserved	BKPSRAMEN	Reserved	

	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			CRCEN	Reserved			PIEN	PHEN	PGEN	PFEN	PEEN	PDEN	PCEN	PBEN	PAEN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	USBHSULPIEN	USBHS ULPI clock enable This bit is set and reset by software. 0: Disabled USBHS ULPI clock 1: Enabled USBHS ULPI clock
29	USBHSEN	USBHS clock enable This bit is set and reset by software. 0: Disabled USBHS clock 1: Enabled USBHS clock
28	ENETPTPEN	Ethernet PTP clock enable This bit is set and reset by software. 0: Disabled Ethernet PTP clock 1: Enabled Ethernet PTP clock
27	ENETRXEN	Ethernet RX clock enable This bit is set and reset by software. 0: Disabled Ethernet RX clock 1: Enabled Ethernet RX clock
26	ENETTXEN	Ethernet TX clock enable This bit is set and reset by software. 0: Disabled Ethernet TX clock 1: Enabled Ethernet TX clock
25	ENETEN	Ethernet clock enable This bit is set and reset by software. 0: Disabled Ethernet clock 1: Enabled Ethernet clock
24	Reserved	Must be kept at reset value.
23	IPAEN	IPA clock enable This bit is set and reset by software. 0: Disabled IPA clock 1: Enabled IPA clock
22	DMA1EN	DMA1 clock enable This bit is set and reset by software. 0: Disabled DMA1 clock

		1: Enabled DMA1 clock
21	DMA0EN	DMA0 clock enable This bit is set and reset by software. 0: Disabled DMA0 clock 1: Enabled DMA0 clock
20	TCMSRAMEN	TCMSRAM clock enable This bit is set and reset by software. 0: Disabled TCMSRAM clock 1: Enabled TCMSRAM clock
19	Reserved	Must be kept at reset value.
18	BKPSRAMEN	BKPSRAM clock enable This bit is set and reset by software. 0: Disabled BKPSRAM clock 1: Enabled BKPSRAM clock
17:13	Reserved	Must be kept at reset value.
12	CRCEN	CRC clock enable This bit is set and reset by software. 0: Disabled CRC clock 1: Enabled CRC clock
11:9	Reserved	Must be kept at reset value.
8	PIEN	GPIO port I clock enable This bit is set and reset by software. 0: Disabled GPIO port I clock 1: Enabled GPIO port I clock
7	PHEN	GPIO port H clock enable This bit is set and reset by software. 0: Disabled GPIO port H clock 1: Enabled GPIO port H clock
6	PGEN	GPIO port G clock enable This bit is set and reset by software. 0: Disabled GPIO port G clock 1: Enabled GPIO port G clock
5	PFEN	GPIO port F clock enable This bit is set and reset by software. 0: Disabled GPIO port F clock 1: Enabled GPIO port F clock
4	PEEN	GPIO port E clock enable This bit is set and reset by software.

		0: Disabled GPIO port E clock 1: Enabled GPIO port E clock
3	PDEN	GPIO port D clock enable This bit is set and reset by software. 0: Disabled GPIO port D clock 1: Enabled GPIO port D clock
2	PCEN	GPIO port C clock enable This bit is set and reset by software. 0: Disabled GPIO port C clock 1: Enabled GPIO port C clock
1	PBEN	GPIO port B clock enable This bit is set and reset by software. 0: Disabled GPIO port B clock 1: Enabled GPIO port B clock
0	PAEN	GPIO port A clock enable This bit is set and reset by software. 0: Disabled GPIO port A clock 1: Enabled GPIO port A clock

5.3.11. AHB2 enable register (RCU_AHB2EN)

Address offset: 0x34

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								USBFSEN	TRNGEN	HAUEN	CAUEN	PKCAUEN	Reserved		DCIEN
								rw	rw	rw	rw	rw			rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	USBFSEN	USBFS clock enable This bit is set and reset by software. 0: Disabled USBFS clock 1: Enabled USBFS clock
6	TRNGEN	TRNG clock enable This bit is set and reset by software. 0: Disabled TRNG clock

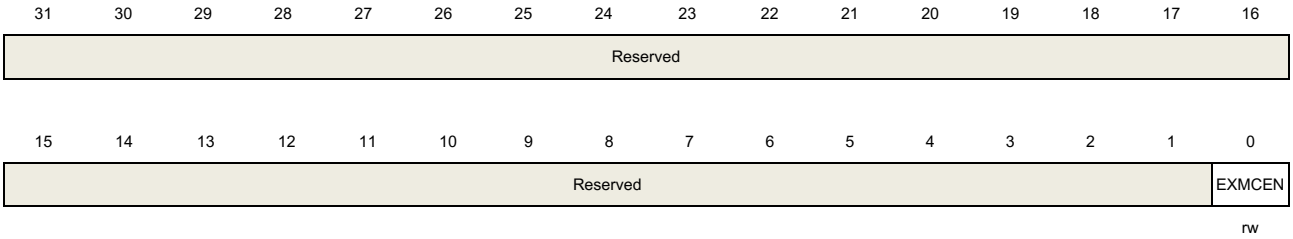
		1: Enabled TRNG clock
5	HAUEN	HAU clock enable This bit is set and reset by software. 0: Disabled HAU clock 1: Enabled HAU clock
4	CAUEN	CAU clock enable This bit is set and reset by software. 0: Disabled CAU clock 1: Enabled CAU clock
3	PKCAUEN	PKCAU clock enable This bit is set and reset by software. 0: Disabled PKCAU clock 1: Enabled PKCAU clock
2:1	Reserved	Must be kept at reset value.
0	DCIEN	DCI clock enable This bit is set and reset by software. 0: Disabled DCI clock 1: Enabled DCI clock

5.3.12. AHB3 enable register (RCU_AHB3EN)

Address offset: 0x38

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	EXMCEN	EXMC clock enable This bit is set and reset by software. 0: Disabled EXMC clock 1: Enabled EXMC clock

5.3.13. 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	PMUEN	Reserved	CAN1EN	CAN0EN	Reserved	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	I2C5EN	I2C4EN	WWDGTEN	I2C3EN	Reserved	TIMER13EN	TIMER12EN	TIMER11EN	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 This bit is set and reset by software. 0: Disabled UART7 clock 1: Enabled UART7 clock
30	UART6EN	UART6 clock enable This bit is set and reset by software. 0: Disabled UART6 clock 1: Enabled UART6 clock
29	DACEN	DAC clock enable This bit is set and reset by software. 0: Disabled DAC clock 1: Enabled DAC clock
28	PMUEN	PMU clock enable This bit is set and reset by software. 0: Disabled PMU clock 1: Enabled PMU clock
27	Reserved	Must be kept at reset value.
26	CAN1EN	CAN1 clock enable This bit is set and reset by software. 0: Disabled CAN1 clock 1: Enabled CAN1 clock
25	CAN0EN	CAN0 clock enable This bit is set and reset by software. 0: Disabled CAN0 clock 1: Enabled CAN0 clock

24	Reserved	Must be kept at reset value.
23	I2C2EN	I2C2 clock enable This bit is set and reset by software. 0: Disabled I2C2 clock 1: Enabled I2C2 clock
22	I2C1EN	I2C1 clock enable This bit is set and reset by software. 0: Disabled I2C1 clock 1: Enabled I2C1 clock
21	I2C0EN	I2C0 clock enable This bit is set and reset by software. 0: Disabled I2C0 clock 1: Enabled I2C0 clock
20	UART4EN	UART4 clock enable This bit is set and reset by software. 0: Disabled UART4 clock 1: Enabled UART4 clock
19	UART3EN	UART3 clock enable This bit is set and reset by software. 0: Disabled UART3 clock 1: Enabled UART3 clock
18	USART2EN	USART2 clock enable This bit is set and reset by software. 0: Disabled USART2 clock 1: Enabled USART2 clock
17	USART1EN	USART1 clock enable This bit is set and reset by software. 0: Disabled USART1 clock 1: Enabled USART1 clock
16	Reserved	Must be kept at reset value.
15	SPI2EN	SPI2 clock enable This bit is set and reset by software. 0: Disabled SPI2 clock 1: Enabled SPI2 clock
14	SPI1EN	SPI1 clock enable This bit is set and reset by software. 0: Disabled SPI1 clock 1: Enabled SPI1 clock
13	I2C5EN	I2C5 clock enable

This bit is set and reset by software.
0: Disabled I2C5 clock
1: Enabled I2C5 clock

12	I2C4EN	I2C4 clock enable This bit is set and reset by software. 0: Disabled I2C4 clock 1: Enabled I2C4 clock
11	WWDGTEN	WWDGT clock enable This bit is set and reset by software. 0: Disabled WWDGT clock 1: Enabled WWDGT clock
10	I2C3EN	I2C3 clock enable This bit is set and reset by software. 0: Disabled I2C3 clock 1: Enabled I2C3 clock
9	Reserved	Must be kept at reset value.
8	TIMER13EN	TIMER13 clock enable This bit is set and reset by software. 0: Disabled TIMER13 clock 1: Enabled TIMER13 clock
7	TIMER12EN	TIMER12 clock enable This bit is set and reset by software. 0: Disabled TIMER12 clock 1: Enabled TIMER12 clock
6	TIMER11EN	TIMER11 clock enable This bit is set and reset by software. 0: Disabled TIMER11 clock 1: Enabled TIMER11 clock
5	TIMER6EN	TIMER6 clock enable This bit is set and reset by software. 0: Disabled TIMER6 clock 1: Enabled TIMER6 clock
4	TIMER5EN	TIMER5 clock enable This bit is set and reset by software. 0: Disabled TIMER5 clock 1: Enabled TIMER5 clock
3	TIMER4EN	TIMER4 clock enable This bit is set and reset by software. 0: Disabled TIMER4 clock

		1: Enabled TIMER4 clock
2	TIMER3EN	TIMER3 clock enable This bit is set and reset by software. 0: Disabled TIMER3 clock 1: Enabled TIMER3 clock
1	TIMER2EN	TIMER2 clock enable This bit is set and reset by software. 0: Disabled TIMER2 clock 1: Enabled TIMER2 clock
0	TIMER1EN	TIMER1 clock enable This bit is set and reset by software. 0: Disabled TIMER1 clock 1: Enabled TIMER1 clock

5.3.14. 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
Reserved					TLIEN	Reserved			SAIEN	SPI5EN	SPI4EN	Reserved	TIMER10EN	TIMER9EN	TIMER8EN
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SYSCFGEN	SPI3EN	SPI0EN	SDIOEN	ADC2EN	ADC1EN	ADC0EN	Reserved		USART5EN	USART0EN	Reserved		TIMER7EN	TIMER0EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	TLIEN	TLI clock enable This bit is set and reset by software. 0: Disabled TLI clock 1: Enabled TLI clock
25:23	Reserved	Must be kept at reset value.
22	SAIEN	SAI clock enable This bit is set and reset by software. 0: Disabled SAI clock 1: Enabled SAI clock
21	SPI5EN	SPI5 clock enable

		This bit is set and reset by software. 0: Disabled SPI5 clock 1: Enabled SPI5 clock
20	SPI4EN	SPI4 clock enable This bit is set and reset by software. 0: Disabled SPI4 clock 1: Enabled SPI4 clock
19	Reserved	Must be kept at reset value.
18	TIMER10EN	TIMER10 clock enable This bit is set and reset by software. 0: Disabled TIMER10 clock 1: Enabled TIMER10 clock
17	TIMER9EN	TIMER9 clock enable This bit is set and reset by software. 0: Disabled TIMER9 clock 1: Enabled TIMER9 clock
16	TIMER8EN	TIMER8 clock enable This bit is set and reset by software. 0: Disabled TIMER8 clock 1: Enabled TIMER8 clock
15	Reserved	Must be kept at reset value.
14	SYSCFGEN	SYSCFG clock enable This bit is set and reset by software. 0: Disabled SYSCFG clock 1: Enabled SYSCFG clock
13	SPI3EN	SPI3 clock enable This bit is set and reset by software. 0: Disabled SPI3 clock 1: Enabled SPI3 clock
12	SPI0EN	SPI0 clock enable This bit is set and reset by software. 0: Disabled SPI0 clock 1: Enabled SPI0 clock
11	SDIOEN	SDIO clock enable This bit is set and reset by software. 0: Disabled SDIO clock 1: Enabled SDIO clock
10	ADC2EN	ADC2 clock enable This bit is set and reset by software.

		0: Disabled ADC2 clock 1: Enabled ADC2 clock
9	ADC1EN	ADC1 clock enable This bit is set and reset by software. 0: Disabled ADC1 clock 1: Enabled ADC1 clock
8	ADC0EN	ADC0 clock enable This bit is set and reset by software. 0: Disabled ADC0 clock 1: Enabled ADC0 clock
7:6	Reserved	Must be kept at reset value.
5	USART5EN	USART5 clock enable This bit is set and reset by software. 0: Disabled USART5 clock 1: Enabled USART5 clock
4	USART0EN	USART0 clock enable This bit is set and reset by software. 0: Disabled USART0 clock 1: Enabled USART0 clock
3:2	Reserved	Must be kept at reset value.
1	TIMER7EN	TIMER7 clock enable This bit is set and reset by software. 0: Disabled TIMER7 clock 1: Enabled TIMER7 clock
0	TIMER0EN	TIMER0 clock enable This bit is set and reset by software. 0: Disabled TIMER0 clock 1: Enabled TIMER0 clock

5.3.15. AHB1 sleep mode enable register (RCU_AHB1SPEN)

Address offset: 0x50

Reset value: 0x7EEF 91FF

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	USBHSULPISPEN	USBHSSPEN	ENETPTPSPEN	ENETRXSPEN	ENETTXSPEN	ENETSPEN	Reserved	IPASPEN	DMA1SPEN	DMA0SPEN	Reserved	SRAM2SPEN	BKPSRAMSPEN	SRAM1SPEN	SRAM0SPEN
	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

FMCSPEN	Reserved	CRCSPEN	Reserved	PISPEN	PHSPEN	PGSPEN	PFSPEN	PESPEN	PDSPEN	PCSPEN	PBSPEN	PASPEN
rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	USBHSULPISPEN	USBHS ULPI clock enable when sleep mode This bit is set and reset by software. 0: Disabled USBHS ULPI clock when sleep mode 1: Enabled USBHS ULPI clock when sleep mode
29	USBHSSPEN	USBHS clock enable when sleep mode This bit is set and reset by software. 0: Disabled USBHS clock when sleep mode 1: Enabled USBHS clock when sleep mode
28	ENETPTPSPEN	Ethernet PTP clock enable when sleep mode This bit is set and reset by software. 0: Disabled Ethernet PTP clock when sleep mode 1: Enabled Ethernet PTP clock when sleep mode
27	ENETRXSPEN	Ethernet RX clock enable when sleep mode This bit is set and reset by software. 0: Disabled Ethernet RX clock when sleep mode 1: Enabled Ethernet RX clock when sleep mode
26	ENETTXSPEN	Ethernet TX clock enable when sleep mode This bit is set and reset by software. 0: Disabled Ethernet TX clock when sleep mode 1: Enabled Ethernet TX clock when sleep mode
25	ENETSPEN	Ethernet clock enable when sleep mode This bit is set and reset by software. 0: Disabled Ethernet clock when sleep mode 1: Enabled Ethernet clock when sleep mode
24	Reserved	Must be kept at reset value.
23	IPASPEN	IPA clock enable when sleep mode This bit is set and reset by software. 0: Disabled IPA clock when sleep mode 1: Enabled IPA clock when sleep mode
22	DMA1SPEN	DMA1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled DMA1 clock when sleep mode 1: Enabled DMA1 clock when sleep mode
21	DMA0SPEN	DMA0 clock enable when sleep mode

		This bit is set and reset by software. 0: Disabled DMA0 clock when sleep mode 1: Enabled DMA0 clock when sleep mode
20	Reserved	Must be kept at reset value.
19	SRAM2SPEN	SRAM2 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SRAM2 clock when sleep mode 1: Enabled SRAM2 clock when sleep mode
18	BKPSRAMSPEN	BKPSRAM clock enable when sleep mode This bit is set and reset by software. 0: Disabled BKPSRAM clock when sleep mode 1: Enabled BKPSRAM clock when sleep mode
17	SRAM1SPEN	SRAM1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SRAM1 clock when sleep mode 1: Enabled SRAM1 clock when sleep mode
16	SRAM0SPEN	SRAM0 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SRAM0 clock when sleep mode 1: Enabled SRAM0 clock when sleep mode
15	FMCSPEN	FMC clock enable when sleep mode This bit is set and reset by software. 0: Disabled FMC clock when sleep mode 1: Enabled FMC clock when sleep mode
14:13	Reserved	Must be kept at reset value.
12	CRCSPEN	CRC clock enable when sleep mode This bit is set and reset by software. 0: Disabled CRC clock when sleep mode 1: Enabled CRC clock when sleep mode
11:9	Reserved	Must be kept at reset value.
8	PISPEN	GPIO port I clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port I clock when sleep mode 1: Enabled GPIO port I clock when sleep mode
7	PHSPEN	GPIO port H clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port H clock when sleep mode 1: Enabled GPIO port H clock when sleep mode

6	PGSPEN	GPIO port G clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port G clock when sleep mode 1: Enabled GPIO port G clock when sleep mode
5	PFSPEN	GPIO port F clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port F clock when sleep mode 1: Enabled GPIO port F clock when sleep mode
4	PESPEN	GPIO port E clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port E clock when sleep mode 1: Enabled GPIO port E clock when sleep mode
3	PDSPEN	GPIO port D clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port D clock when sleep mode 1: Enabled GPIO port D clock when sleep mode
2	PCSPEN	GPIO port C clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port C clock when sleep mode 1: Enabled GPIO port C clock when sleep mode
1	PBSPEN	GPIO port B clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port B clock when sleep mode 1: Enabled GPIO port B clock when sleep mode
0	PASPEN	GPIO port A clock enable when sleep mode This bit is set and reset by software. 0: Disabled GPIO port A clock when sleep mode 1: Enabled GPIO port A clock when sleep mode

5.3.16. AHB2 sleep mode enable register (RCU_AHB2SPEN)

Address offset: 0x54

Reset value: 0x0000 00F9

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								USBFSSPEN	TRNGSPEN	HAUSPEN	CAUSPEN	PKCAUSPEN	Reserved		DCISPEN

rw rw rw rw rw rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	USBFSSPEN	USBFS clock enable when sleep mode This bit is set and reset by software. 0: Disabled USBFS clock when sleep mode 1: Enabled USBFS clock when sleep mode
6	TRNGSPEN	TRNG clock enable when sleep mode This bit is set and reset by software. 0: Disabled TRNG clock when sleep mode 1: Enabled TRNG clock when sleep mode
5	HAUSPEN	HAU clock enable when sleep mode This bit is set and reset by software. 0: Disabled HAU clock when sleep mode 1: Enabled HAU clock when sleep mode
4	CAUSPEN	CAU clock enable when sleep mode This bit is set and reset by software. 0: Disabled CAU clock when sleep mode 1: Enabled CAU clock when sleep mode
3	PKCAUSPEN	PKCAU clock enable when sleep mode This bit is set and reset by software. 0: Disabled PKCAU clock when sleep mode 1: Enabled PKCAU clock when sleep mode
2:1	Reserved	Must be kept at reset value.
0	DCISPEN	DCI clock enable when sleep mode This bit is set and reset by software. 0: Disabled DCI clock when sleep mode 1: Enabled DCI clock when sleep mode

5.3.17. AHB3 sleep mode enable register (RCU_AHB3SPEN)

Address offset: 0x58

Reset value: 0x0000 0001

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	EXMCSPEN
	rw

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	EXMCSPEN	EXMC clock enable when sleep mode This bit is set and reset by software. 0: Disabled EXMC clock when sleep mode 1: Enabled EXMC clock when sleep mode

5.3.18. APB1 sleep mode enable register (RCU_APB1SPEN)

Address offset: 0x60

Reset value: 0xF6FE FDFF

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	PMUSPEN	Reserved	CAN1SPEN	CAN0SPEN	Reserved	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	I2C5SPEN	I2C4SPEN	WWDGTSPEN	I2C3SPEN	Reserved	TIMER13SPEN	TIMER12SPEN	TIMER11SPEN	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 This bit is set and reset by software. 0: Disabled UART7 clock when sleep mode 1: Enabled UART7 clock when sleep mode
30	UART6SPEN	UART6 clock enable when sleep mode This bit is set and reset by software. 0: Disabled UART6 clock when sleep mode 1: Enabled UART6 clock when sleep mode
29	DACSPEN	DAC clock enable when sleep mode This bit is set and reset by software. 0: Disabled DAC clock when sleep mode 1: Enabled DAC clock when sleep mode
28	PMUSPEN	PMU clock enable when sleep mode This bit is set and reset by software. 0: Disabled PMU clock when sleep mode

		1: Enabled PMU clock when sleep mode
27	Reserved	Must be kept at reset value.
26	CAN1SPEN	CAN1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled CAN1 clock when sleep mode 1: Enabled CAN1 clock when sleep mode
25	CAN0SPEN	CAN0 clock enable when sleep mode This bit is set and reset by software. 0: Disabled CAN0 clock when sleep mode 1: Enabled CAN0 clock when sleep mode
24	Reserved	Must be kept at reset value.
23	I2C2SPEN	I2C2 clock enable when sleep mode This bit is set and reset by software. 0: Disabled I2C2 clock when sleep mode 1: Enabled I2C2 clock when sleep mode
22	I2C1SPEN	I2C1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled I2C1 clock when sleep mode 1: Enabled I2C1 clock when sleep mode
21	I2C0SPEN	I2C0 clock enable when sleep mode This bit is set and reset by software. 0: Disabled I2C0 clock when sleep mode 1: Enabled I2C0 clock when sleep mode
20	UART4SPEN	UART4 clock enable when sleep mode This bit is set and reset by software. 0: Disabled UART4 clock when sleep mode 1: Enabled UART4 clock when sleep mode
19	UART3SPEN	UART3 clock enable when sleep mode This bit is set and reset by software. 0: Disabled UART3 clock when sleep mode 1: Enabled UART3 clock when sleep mode
18	USART2SPEN	USART2 clock enable when sleep mode This bit is set and reset by software. 0: Disabled USART2 clock when sleep mode 1: Enabled USART2 clock when sleep mode
17	USART1SPEN	USART1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled USART1 clock when sleep mode

		1: Enabled USART1 clock when sleep mode
16	Reserved	Must be kept at reset value.
15	SPI2SPEN	SPI2 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SPI2 clock when sleep mode 1: Enabled SPI2 clock when sleep mode
14	SPI1SPEN	SPI1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SPI1 clock when sleep mode 1: Enabled SPI1 clock when sleep mode
13	I2C5SPEN	I2C5 clock enable when sleep mode This bit is set and reset by software. 0: Disabled I2C5 clock when sleep mode 1: Enabled I2C5 clock when sleep mode
12	I2C4SPEN	I2C4 clock enable when sleep mode This bit is set and reset by software. 0: Disabled I2C4 clock when sleep mode 1: Enabled I2C4 clock when sleep mode
11	WWDGTSPEN	WWDGT clock enable when sleep mode This bit is set and reset by software. 0: Disabled WWDGT clock when sleep mode 1: Enabled WWDGT clock when sleep mode
10	I2C3SPEN	I2C3 clock enable when sleep mode This bit is set and reset by software. 0: Disabled I2C3 clock when sleep mode 1: Enabled I2C3 clock when sleep mode
9	Reserved	Must be kept at reset value.
8	TIMER13SPEN	TIMER13 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER13 clock when sleep mode 1: Enabled TIMER13 clock when sleep mode
7	TIMER12SPEN	TIMER12 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER12 clock when sleep mode 1: Enabled TIMER12 clock when sleep mode
6	TIMER11SPEN	TIMER11 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER11 clock when sleep mode

		1: Enabled TIMER11 clock when sleep mode
5	TIMER6SPEN	TIMER6 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER6 clock when sleep mode 1: Enabled TIMER6 clock when sleep mode
4	TIMER5SPEN	TIMER5 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER5 clock when sleep mode 1: Enabled TIMER5 clock when sleep mode
3	TIMER4SPEN	TIMER4 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER4 clock when sleep mode 1: Enabled TIMER4 clock when sleep mode
2	TIMER3SPEN	TIMER3 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER3 clock when sleep mode 1: Enabled TIMER3 clock when sleep mode
1	TIMER2SPEN	TIMER2 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER2 clock when sleep mode 1: Enabled TIMER2 clock when sleep mode
0	TIMER1SPEN	TIMER1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER1 clock when sleep mode 1: Enabled TIMER1 clock when sleep mode

5.3.19. APB2 sleep mode enable register (RCU_APB2SPEN)

Address offset: 0x64

Reset value: 0x0477 7F33

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					TLISPEN	Reserved			SAISPEN	SPI5SPEN	SPI4SPEN	Reserved	TIMER10SPEN	TIMER9SPEN	TIMER8SPEN
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SYSCFGSPEN	SPI3SPEN	SPI0SPEN	SDIOSPEN	ADC2SPEN	ADC1SPEN	ADC0SPEN	Reserved		USART5SPEN	USART0SPEN	Reserved		TIMER7SPEN	TIMER0SPEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	TLISPEN	TLI clock enable when sleep mode This bit is set and reset by software. 0: Disabled TLI clock when sleep mode 1: Enabled TLI clock when sleep mode
25:23	Reserved	Must be kept at reset value.
22	SAISPEN	SAI clock enable when sleep mode This bit is set and reset by software. 0: Disabled SAI clock when sleep mode 1: Enabled SAI clock when sleep mode
21	SPI5SPEN	SPI5 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SPI5 clock when sleep mode 1: Enabled SPI5 clock when sleep mode
20	SPI4SPEN	SPI4 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SPI4 clock when sleep mode 1: Enabled SPI4 clock when sleep mode
19	Reserved	Must be kept at reset value.
18	TIMER10SPEN	TIMER10 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER10 clock when sleep mode 1: Enabled TIMER10 clock when sleep mode
17	TIMER9SPEN	TIMER9 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER9 clock when sleep mode 1: Enabled TIMER9 clock when sleep mode
16	TIMER8SPEN	TIMER8 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER8 clock when sleep mode 1: Enabled TIMER8 clock when sleep mode
15	Reserved	Must be kept at reset value.
14	SYSCFGSPEN	SYSCFG clock enable when sleep mode This bit is set and reset by software. 0: Disabled SYSCFG clock when sleep mode 1: Enabled SYSCFG clock when sleep mode
13	SPI3SPEN	SPI3 clock enable when sleep mode

		This bit is set and reset by software. 0: Disabled SPI3 clock when sleep mode 1: Enabled SPI3 clock when sleep mode
12	SPI0SPEN	SPI0 clock enable when sleep mode This bit is set and reset by software. 0: Disabled SPI0 clock when sleep mode 1: Enabled SPI0 clock when sleep mode
11	SDIOSPEN	SDIO clock enable when sleep mode This bit is set and reset by software. 0: Disabled SDIO clock when sleep mode 1: Enabled SDIO clock when sleep mode
10	ADC2SPEN	ADC2 clock enable when sleep mode This bit is set and reset by software. 0: Disabled ADC2 clock when sleep mode 1: Enabled ADC2 clock when sleep mode
9	ADC1SPEN	ADC1 clock enable when sleep mode This bit is set and reset by software. 0: Disabled ADC1 clock when sleep mode 1: Enabled ADC1 clock when sleep mode
8	ADC0SPEN	ADC0 clock enable when sleep mode This bit is set and reset by software. 0: Disabled ADC0 clock when sleep mode 1: Enabled ADC0 clock when sleep mode
7:6	Reserved	Must be kept at reset value.
5	USART5SPEN	USART5 clock enable when sleep mode This bit is set and reset by software. 0: Disabled USART5 clock when sleep mode 1: Enabled USART5 clock when sleep mode
4	USART0SPEN	USART0 clock enable when sleep mode This bit is set and reset by software. 0: Disabled USART0 clock when sleep mode 1: Enabled USART0 clock when sleep mode
3:2	Reserved	Must be kept at reset value.
1	TIMER7SPEN	TIMER7 clock enable when sleep mode This bit is set and reset by software. 0: Disabled TIMER7 clock when sleep mode 1: Enabled TIMER7 clock when sleep mode
0	TIMER0SPEN	TIMER0 clock enable when sleep mode This bit is set and reset by software.

0: Disabled TIMER0 clock when sleep mode

1: Enabled TIMER0 clock when sleep mode

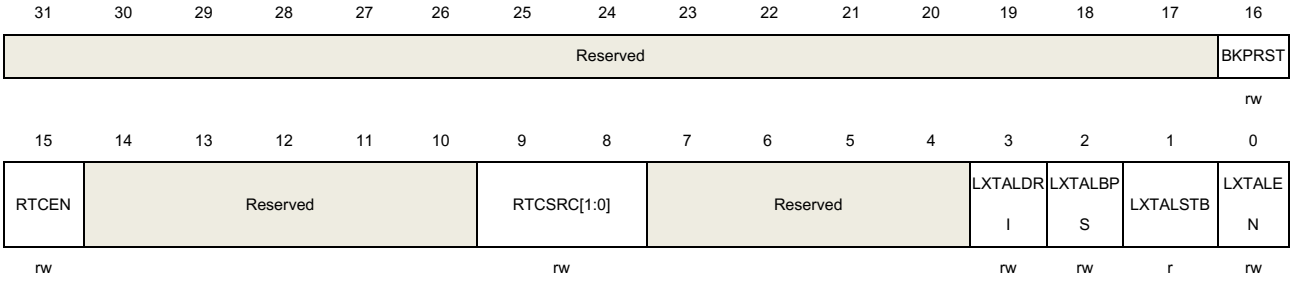
5.3.20. Backup domain control register (RCU_BDCTL)

Address offset: 0x70

Reset value: 0x0000 0008, reset by Backup domain Reset.

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.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit).



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	BKPRST	Backup domain reset This bit is set and reset by software. 0: No reset 1: Resets Backup domain
15	RTCEN	RTC clock enable This bit is set and reset by software. 0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value.
9:8	RTCSRC[1:0]	RTC clock entry selection 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. 00: No clock selected 01: CK_LXTAL selected as RTC source clock 10: CK_IRC32K selected as RTC source clock 11: (CK_HXTAL / RTCDIV) selected as RTC source clock, please refer to RTCDIV bits in RCU_CFG0 register.
7:4	Reserved	Must be kept at reset value.

3	LXTALDRI	LXTAL drive capability Set and reset by software. Backup domain reset resets this value. 0: lower driving capability 1: higher driving capability (reset value) **Note:** The LXTALDRI is not in bypass mode.
2	LXTALBPS	LXTAL bypass mode enable Set and reset by software. 0: Disable the LXTAL Bypass mode 1: Enable the LXTAL Bypass mode
1	LXTALSTB	Low speed crystal oscillator stabilization flag Set by hardware to indicate if the LXTAL output clock is stable and ready for use. 0: LXTAL is not stable 1: LXTAL is stable
0	LXTALEN	LXTAL enable Set and reset by software. 0: Disable LXTAL 1: Enable LXTAL

5.3.21. 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
LP RSTF	WWDGTR STF	FWDGT RSTF	SW RSTF	POR RSTF	EP RSTF	BOR RSTF	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														IRC32K STB	IRC32KEN
														r	rw

Bits	Fields	Descriptions
31	LPRSTF	Low-power reset flag Set by hardware when Deep-sleep /standby reset generated. Reset by writing 1 to the RSTFC bit. 0: No Low-power management reset generated 1: Low-power management reset generated
30	WWDGTRSTF	Window watchdog timer reset flag Set by hardware when a window watchdog timer reset generated.

		Reset by writing 1 to the RSTFC bit. 0: No window watchdog reset generated 1: Window watchdog reset generated
29	FWDGTRSTF	Free watchdog timer reset flag Set by hardware when a free watchdog timer reset generated. Reset by writing 1 to the RSTFC bit. 0: No free watchdog timer reset generated 1: free Watchdog timer reset generated
28	SWRSTF	Software reset flag Set by hardware when a software reset generated. Reset by writing 1 to the RSTFC bit. 0: No software reset generated 1: Software reset generated
27	PORRSTF	Power reset flag Set by hardware when a Power reset generated. Reset by writing 1 to the RSTFC bit. 0: No Power reset generated 1: Power reset generated
26	EPRSTF	External PIN reset flag Set by hardware when an External PIN reset generated. Reset by writing 1 to the RSTFC bit. 0: No External PIN reset generated 1: External PIN reset generated
25	BORRSTF	BOR reset flag Set by hardware when a BOR reset generated. Reset by writing 1 to the RSTFC bit. 0: No BOR reset generated 1: BOR reset generated
24	RSTFC	Reset flag clear This bit is set by software to clear all reset flags. 0: Not clear reset flags 1: Clear reset flags
23:2	Reserved	Must be kept at reset value.
1	IRC32KSTB	IRC32K stabilization flag Set by hardware to indicate if the IRC32K output clock is stable and ready for use. 0: IRC32K is not stable 1: IRC32K is stable
0	IRC32KEN	IRC32K enable Set and reset by software. 0: Disable IRC32K

1: Enable IRC32K

5.3.22. PLL clock spread spectrum control register (RCU_PLLSSCTL)

Address offset: 0x80

Reset value: 0x0000 0000

The spread spectrum modulation is available only for the main PLL clock.

The RCU_PLLSSCTL register must be written when the main PLL is disabled.

This register is used to configure the PLL spread spectrum clock generation according to the following formulas:

MODCNT = round(f_{PLLIN}/4/f_{mod})

MODSTEP = round(mdamp*PLLN*2^{14}/(MODCNT*100))

Where f_{PLLIN} represents the PLL input clock frequency, f_{mod} represents the spread spectrum modulation frequency, mdamp represents the spread spectrum modulation amplitude expressed as a percentage, PLLN represents the PLL clock frequency multiplication factor.

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
SSCGON	SS_TYPE	Reserved		MODSTEP[14:3]											
rw	rw			rw											

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

Bits	Fields	Descriptions
31	SSCGON	PLL spread spectrum modulation enable 0: Spread spectrum modulation disable 1: Spread spectrum modulation enable
30	SS_TYPE	PLL spread spectrum modulation type select 0: Center spread selected 1: Down spread selected
29:28	Reserved	Must be kept at reset value.
27:13	MODSTEP	These bits configure PLL spread spectrum modulation profile amplitude and frequency. The following criteria must be met: MODSTEP*MODCNT<=2^{15}-1.
12:0	MODCNT	These bits configure PLL spread spectrum modulation profile amplitude and frequency. The following criteria must be met: MODSTEP*MODCNT<=2^{15}-1.

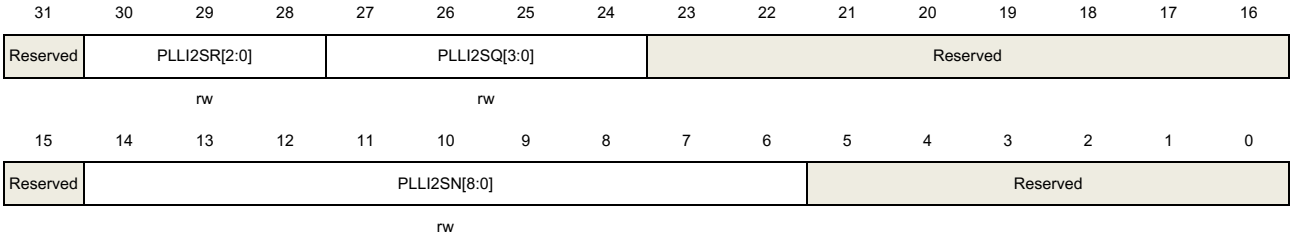
5.3.23. PLLI2S register (RCU_PLLI2S)

Address offset: 0x84

Reset value: 0x2400 3000

To configure the PLLI2S clock, refer to the following formula:
CK_PLLI2SVCOSRC = CK_PLLSRC / PLLPSC
CK_PLLI2SVCO = CK_PLLI2SVCOSRC × PLLI2SN
CK_PLLI2SR = CK_PLLI2SVCO / PLLI2SR To configure the PLLI2S clock, refer to the following formula:
CK_PLLI2SVCOSRC = CK_PLLSRC / PLLPSC
CK_PLLI2SVCO = CK_PLLI2SVCOSRC × PLLI2SN
CK_PLLI2SR = CK_PLLI2SVCO / PLLI2SR

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.
30:28	PLLI2SR[2:0]	The PLLI2S R output frequency division factor from PLLI2S VCO clock Set and reset by software when the PLLI2S is disable. These bits used to generate PLLI2S R output clock (CK_PLLI2SR) from PLLI2S VCO clock (CK_PLLI2SVCO). The CK_PLLI2SR is used to generate I2S clock (<=200MHz). The CK_PLLI2SVCO is described in PLLI2SN bits in RCU_PLLI2S register. 000: Reserved 001: Reserved 010: CK_PLLI2SR = CK_PLLI2SVCO / 2 011: CK_PLLI2SR = CK_PLLI2SVCO / 3 100: CK_PLLI2SR = CK_PLLI2SVCO / 4 … 111: CK_PLLI2SR = CK_PLLI2SVCO / 7
27:24	PLLI2SQ[3:0]	The PLLI2SQ output frequency division factor from PLLI2S VCO clock Set and reset by software when the PLLI2S is disable. These bits used to generate PLLI2SQ output clock (CK_PLLI2SQ) from PLLI2S VCO clock (CK_PLLI2SVCO). The CK_PLLI2SVCO is described in PLLI2SQ bits in RCU_PLLI2S register. 0000: Reserved 0001: Reserved 0010: CK_PLLI2SQ = CK_PLLI2SVCO / 2 0011: CK_PLLI2SQ = CK_PLLI2SVCO / 3 0100: CK_PLLI2SQ = CK_PLLI2SVCO / 4 … 1111: CK_PLLI2SQ = CK_PLLI2SVCO / 15

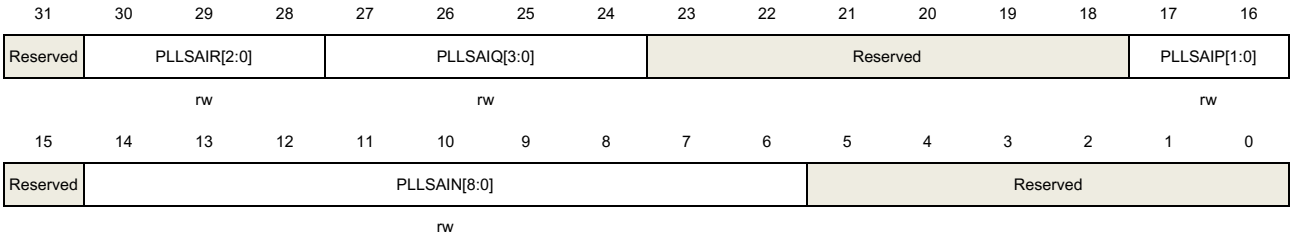
23:15	Reserved	Must be kept at reset value.
14:6	PLLI2SN[8:0]	The PLLI2S VCO clock multiplication factor Set and reset by software (only use word/half-word write) when the PLLI2S is disable. These bits used to generate PLLI2S VCO clock (CK_PLLI2SVCO) from PLLI2S VCO source clock (CK_PLLI2SVCOSRC). The CK_PLLI2SVCOSRC is described in PLLPSC bits in RCU_PLL register Note: The frequency of CK_PLLI2SVCO is between 100MHz to 500MHz The value of PLLI2SN must : 50<=PLLI2SN<=500 000000000: Reserved 000000001: Reserved … 000110001: Reserved 000110010: CK_PLLI2SVCO = CK_PLLI2SVCOSRC x 50 000110011: CK_PLLI2SVCO = CK_PLLI2SVCOSRC x 51 … 111110100: CK_PLLI2SVCO = CK_PLLI2SVCOSRC x 500 111110101: Reserved … 111111111: Reserved
5:0	Reserved	Must be kept at reset value.

5.3.24. PLLSAI register (RCU_PLLSAI)

Address offset: 0x88
Reset value: 0x2400 3000

To configure the PLLSAI clock, refer to the following formula:
CK_PLLSAIVCOSRC = CK_PLLSRC / PLLPSC
CK_PLLSAIVCO = CK_ PLLSAIVCOSRC × PLLSAIN
CK_PLLSAIP = CK_ PLLSAIVCO / PLLSAIP
CK_PLLSAIR = CK_ PLLSAIVCO / PLLSAIR

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.
30:28	PLLSAIR[2:0]	The PLLSAI R output frequency division factor from PLLSAI VCO clock

		Set and reset by software when the PLLSAI is disable. These bits used to generate PLLSAI R output clock (CK_PLLSAIR) from PLLSAI VCO clock (CK_PLLSAIVCO). The CK_PLLSAIR is used to generate TLI clock (<=216MHz). The CK_PLLSAIVCO is described in PLLSAIN bits in RCU_PLLSAI register. 000: Reserved 001: Reserved 010: CK_PLLSAIR = CK_PLLSAIVCO / 2 011: CK_PLLSAIR = CK_PLLSAIVCO / 3 100: CK_PLLSAIR = CK_PLLSAIVCO / 4 … 111: CK_PLLSAIR = CK_PLLSAIVCO / 7
27:24	PLLSAIQ[3:0]	The PLLSAIQ output frequency division factor from PLLSAI VCO clock Set and reset by software when the PLLSAI is disable. These bits used to generate PLLSAIQ output clock (CK_PLLSAIQ) from PLLSAI VCO clock (CK_PLLSAIVCO). The CK_PLLSAIVCO is described in PLLSAIQ bits in RCU_PLLSAI register. 0000: Reserved 0001: Reserved 0010: CK_PLLSAIQ = CK_PLLSAIVCO / 2 0011: CK_PLLSAIQ = CK_PLLSAIVCO / 3 0100: CK_PLLSAIQ = CK_PLLSAIVCO / 4 … 1111: CK_PLLSAIQ = CK_PLLSAIVCO / 15
23:18	Reserved	Must be kept at reset value.
17:16	PLLSAIP[1:0]	The PLLSAI P output frequency division factor from PLLSAI VCO clock Set and reset by software when the PLLSAI is disable. These bits used to generator PLLSAI P output clock (CK_PLLSAIP) from PLLSAI VCO clock (CK_PLLSAIVCO). The CK_PLLSAIP is used to UBSFS/USBHS (48MHz), TRNG (48MHz), or SDIO (<=48MHz). The CK_PLLSAIVCO is described in PLLSAIN bits in RCU_PLLSAI register. 00 : CK_PLLSAIP = CK_PLLSAIVCO / 2 01 : CK_PLLSAIP = CK_PLLSAIVCO / 4 10 : CK_PLLSAIP = CK_PLLSAIVCO / 6 11 : CK_PLLSAIP = CK_PLLSAIVCO / 8
15	Reserved	Must be kept at reset value.
14:6	PLLSAIN[8:0]	The PLLSAI VCO clock multiplication factor Set and reset by software (only use word/half-word write) when the PLLSAI is disable. These bits used to generate PLLSAI VCO clock (CK_PLLSAIVCO) from PLLSAI VCO source clock (CK_PLLSAIVCOSRC). The CK_PLLSAIVCOSRC is described in PLLPSC bits in RCU_PLL register. Note: The frequency of CK_PLLSAIVCO is between 100MHz to 500MHz The value of PLLI2SN must : 50<=PLLSAIN<=500 000000000: Reserved

000000001: Reserved
…
000110001: Reserved
000110010: CK_PLLSAIVCO = CK_PLLSAIVCOSRC x 50
000110011: CK_PLLSAIVCO = CK_PLLSAIVCOSRC x 51
…
111110100: CK_PLLSAIVCO = CK_PLLSAIVCOSRC x 500.
111110101: Reserved
…
111111111: Reserved

5:0	Reserved	Must be kept at reset value.

5.3.25. Clock configuration register 1 (RCU_CFG1)

Address offset: 0x8C
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							TIMERSEL	Reserved		SAISEL[1:0]		Reserved		PLLSAIRDIV[1:0]	
							rw			rw				rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PLLI2SQDIV[4:0]			
												rw			

Bits	Fields	Descriptions
31:25	Reserved	Must be kept at reset value.
24	TIMERSEL	TIMER clock selection This bit is set and reset by software. This bit defined all timer clock selection. 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). 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	Reserved	Must be kept at reset value.
21:20	SAISEL	SAI clock selection

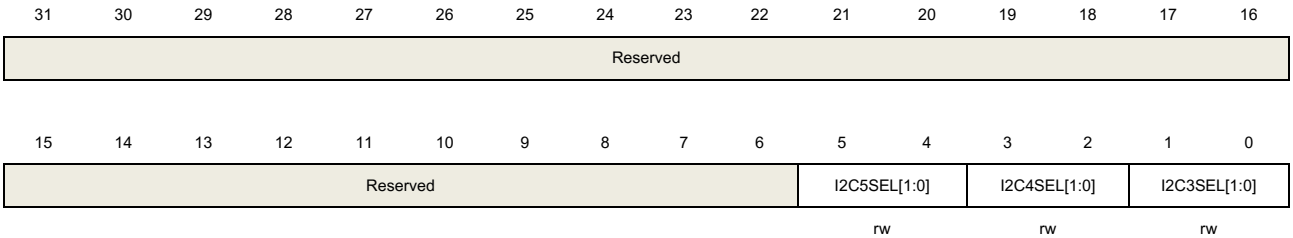
		This bit is set and reset by software. This bit defined all timer clock selection. 00: CK_PLLSAIQ 01: CK_PLLI2SQ 10: I2S_CKIN 11: I2S_CKIN
19:18	Reserved	Must be kept at reset value.
17:16	PLLSAIRDIV[1:0]	The divider factor from PLLSAIR clock These bits are set and reset by software when PLLSAI is disabled. These bits used to generate clock for TLI clock. 00: CK_PLLSAIR / 2 01: CK_PLLSAIR / 4 10: CK_PLLSAIR / 8 11: CK_PLLSAIR / 16
15:5	Reserved	Must be kept at reset value.
4:0	PLLI2SQDIV[4:0]	The divider factor from PLLI2SQ clock These bits are set and reset by software when PLLI2S is disabled. These bits used to generate clock for SAI clock. 00000: CK_PLLI2SQ / 1 00001: CK_PLLI2SQ / 2 00010: CK_PLLI2SQ / 3 … 11111: CK_PLLI2SQ / 32

5.3.26. Clock configuration register 2 (RCU_CFG2)

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
31:6	Reserved	Must be kept at reset value.
5:4	I2C5SEL[1:0]	I2C5 clock source selection Set and reset by software to control the I2C5 clock source. 00: CK_APB1 selected as I2C5 source clock 01: CK_PLLSAIR selected as I2C5 source clock

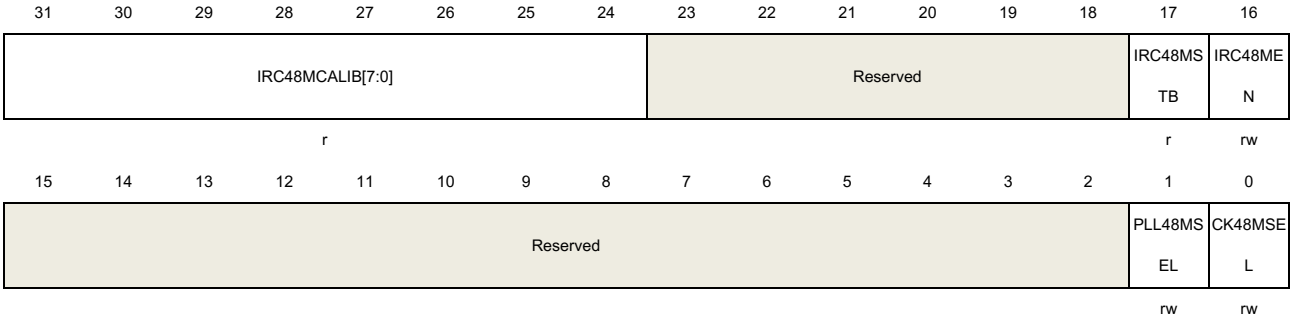
		10: CK_IRC16M selected as I2C5 source clock 11: No select
3:2	I2C4SEL[1:0]	I2C4 clock source selection Set and reset by software to control the I2C4 clock source. 00: CK_APB1 selected as I2C4 source clock 01: CK_PLLSAIR selected as I2C4 source clock 10: CK_IRC16M selected as I2C4 source clock 11: No select
1:0	I2C3SEL[1:0]	I2C3 clock source selection Set and reset by software to control the I2C3 clock source. 00: CK_APB1 selected as I2C3 source clock 01: CK_PLLSAIR selected as I2C3 source clock 10: CK_IRC16M selected as I2C3 source clock 11: No select

5.3.27. Additional clock control register (RCU_ADDCTL)

Address offset: 0xC0

Reset value: 0xXX00 0000 where X is undefined

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 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 Set by hardware to indicate if the IRC48M oscillator is stable and ready for use. 0: IRC48M is not stable 1: IRC48M is stable
16	IRC48MEN	Internal 48MHz RC oscillator enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: IRC48M disable

		1: IRC48M enable
15:2	Reserved	Must be kept at reset value.
1	PLL48MSEL	PLL48M clock selection Set and reset by software. This bit used to generate PLL48M clock which select CK_PLLQ or CK_PLLSAIP clock. 0: Select CK_PLLQ clock 1: Select CK_PLLSAIP clock
0	CK48MSEL	48MHz clock selection 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/SDIO/USBFS/USBHS. The PLL48M clock refer to PLL48MSEL bit in RCU_ADDCTL register. 0: Don't select IRC48M clock(use CK_PLLQ clock or CK_PLLSAIP clock select by PLL48MSEL) 1: Select IRC48M clock

5.3.28. 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)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									IRC48MSTBIC	Reserved					
									w						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	IRC48MSTBIE	Reserved							IRC48MSTBIF	Reserved					
	rw								r						

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22	IRC48MSTBIC	Internal 48 MHz RC oscillator Stabilization interrupt clear Write 1 by software to reset the IRC48MSTBIF flag. 0: Not reset IRC48MSTBIF flag 1: Reset IRC48MSTBIF flag
21:15	Reserved	Must be kept at reset value.
14	IRC48MSTBIE	Internal 48 MHz RC oscillator Stabilization interrupt enable Set and reset by software to enable/disable the IRC48M stabilization interrupt 0: Disable the IRC48M stabilization interrupt

		1: Enable the IRC48M stabilization interrupt
13:7	Reserved	Must be kept at reset value.
6	IRC48MSTBIF	IRC48M stabilization interrupt flag Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set. Reset by software when setting the IRC48MSTBIC bit. 0: No IRC48M stabilization interrupt generated 1: IRC48M stabilization interrupt generated
5:0	Reserved	Must be kept at reset value.

5.3.29. APB1 additional reset register (RCU_ADDAPB1RST)

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
IREFRST	Reserved			CTCRST	Reserved										
rw				rw											

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

Bits	Fields	Descriptions
31	IREFRST	IREF reset This bit is set and reset by software. 0: No reset 1: Reset IREF unit
30:28	Reserved	Must be kept at reset value.
27	CTCRST	CTC reset This bit is set and reset by software. 0: No reset 1: Reset CTC
26:0	Reserved	Must be kept at reset value.

5.3.30. APB1 additional enable register (RCU_ADDAPB1EN)

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
IREFEN	Reserved			CTCEN	Reserved										
rw				rw											

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

Bits	Fields	Descriptions
31	IREFEN	IREF interface clock enable This bit is set and reset by software. 0: Disabled IREF clock 1: Enabled IREF clock
30:28	Reserved	Must be kept at reset value.
27	CTCEN	CTC clock enable This bit is set and reset by software. 0: Disabled CTC clock 1: Enabled CTC clock
26:0	Reserved	Must be kept at reset value.

5.3.31. APB1 additional sleep mode enable register (RCU_ADDAPB1SPEN)

Address offset: 0xE8

Reset value: 0x8800 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
IREF SPEN	Reserved			CTCSP EN	Reserved										
rw				rw											

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

Bits	Fields	Descriptions
31	IREFSPEN	IREF interface clock enable when sleep mode This bit is set and reset by software. 0: Disabled IREF clock when sleep mode 1: Enabled IREF clock when sleep mode
30:28	Reserved	Must be kept at reset value.
27	CTCSPEN	CTC clock enable when sleep mode This bit is set and reset by software 0: Disabled CTC clock when sleep mode

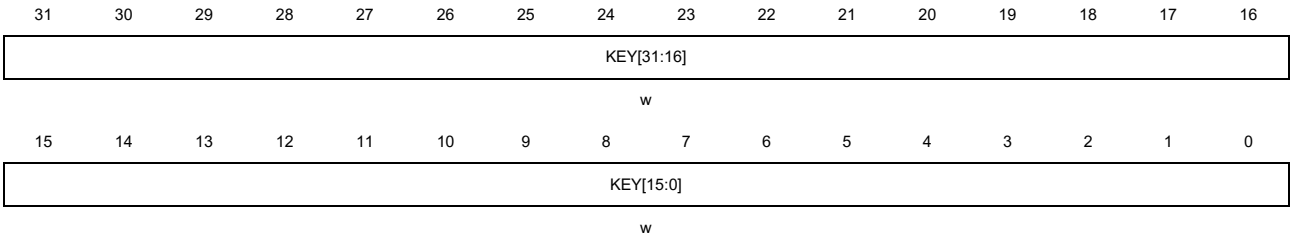
		1: Enabled CTC clock when sleep mode
26:0	Reserved	Must be kept at reset value.

5.3.32. Voltage key register (RCU_VKEY)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



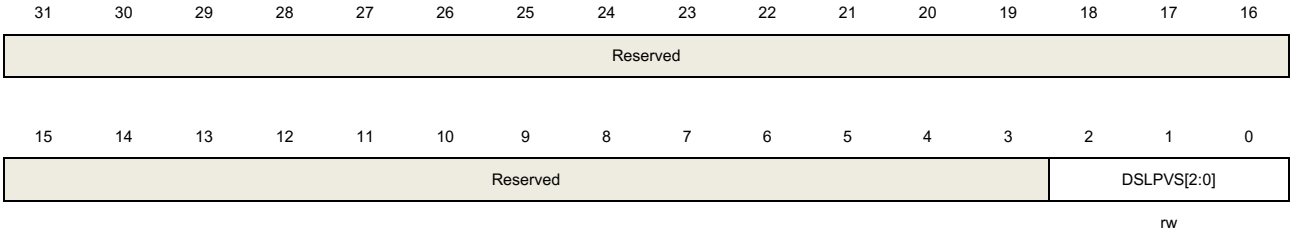
Bits	Fields	Descriptions
31:0	KEY[31:0]	The key of RCU_DSV register These bits are written only by software and read as 0. Only after write 0x1A2B3C4D to the RCU_VKEY, the RCU_DSV register can be written.

5.3.33. Deep-sleep mode voltage register (RCU_DSV)

Address offset: 0x134

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.
2:0	DSLPVS[2:0]	Deep-sleep mode voltage select These bits are set and reset by software. 000: The core voltage is default value in Deep-sleep mode 001: The core voltage is (default value-0.1)V in Deep-sleep mode(customers are not recommended to use it) 010: The core voltage is (default value-0.2)V in Deep-sleep mode(customers are not recommended to use it)

011: The core voltage is (default value-0.3)V in Deep-sleep mode(customers are not recommended to use it)

100~111: Reserved

6. Clock trim controller (CTC)

6.1. Overview

The clock trim controller (CTC) is used to trim internal 48MHz RC oscillator (IRC48M) automatically by hardware. The CTC unit trims the frequency of the IRC48M which is based on an external accurate reference signal source. It can adjust the calibration value to provide a precise IRC48M clock automatically or manually.

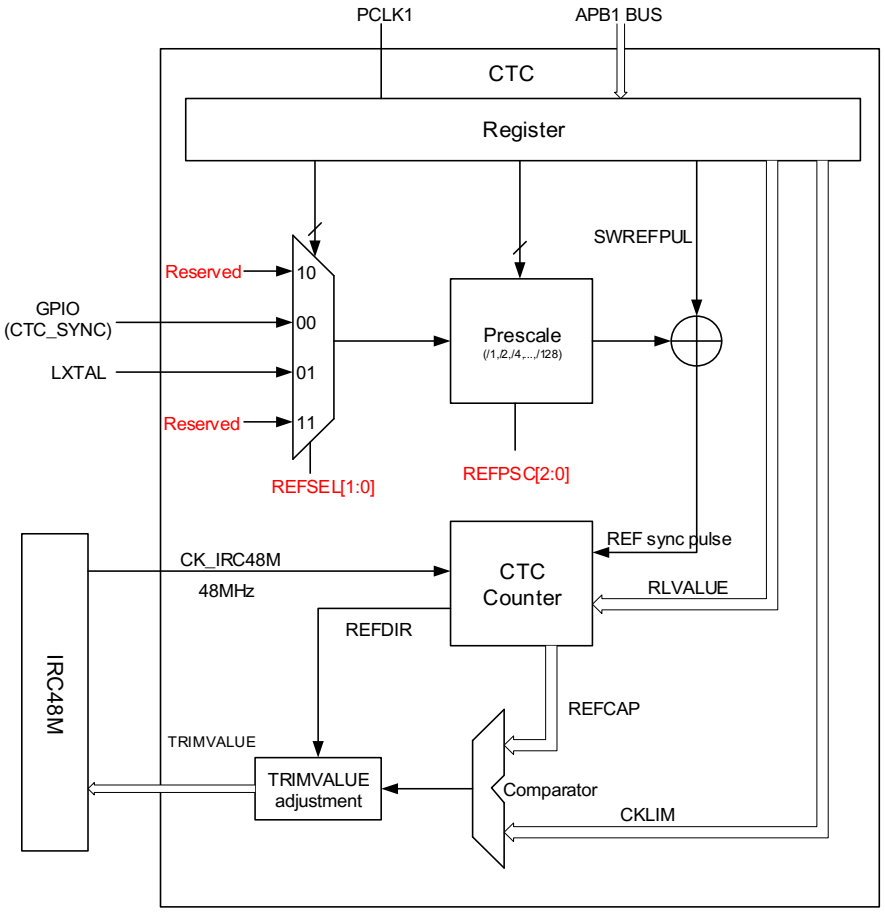
6.2. Characteristics

- Two external reference signal sources: GPIO (CTC_SYNC), LXTAL clock.
- Provide software reference sync pulse.
- Trimmed by hardware without any software action automatically.
- 16 bits trim counter with reference signal source capture and reload function.
- 8 bits clock trim base value used for frequency evaluation and automatic trim.
- Flags or interrupts to indicate whether the clock trim status is OK (CKOKIF), warning (CKWARNIF) or error (ERRIF).

6.3. Function overview

Figure 6-1, Block diagram of CTC provides details on the internal configuration of the CTC.

Figure 6-1. Block diagram of CTC



6.3.1. Reference 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 uses 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.

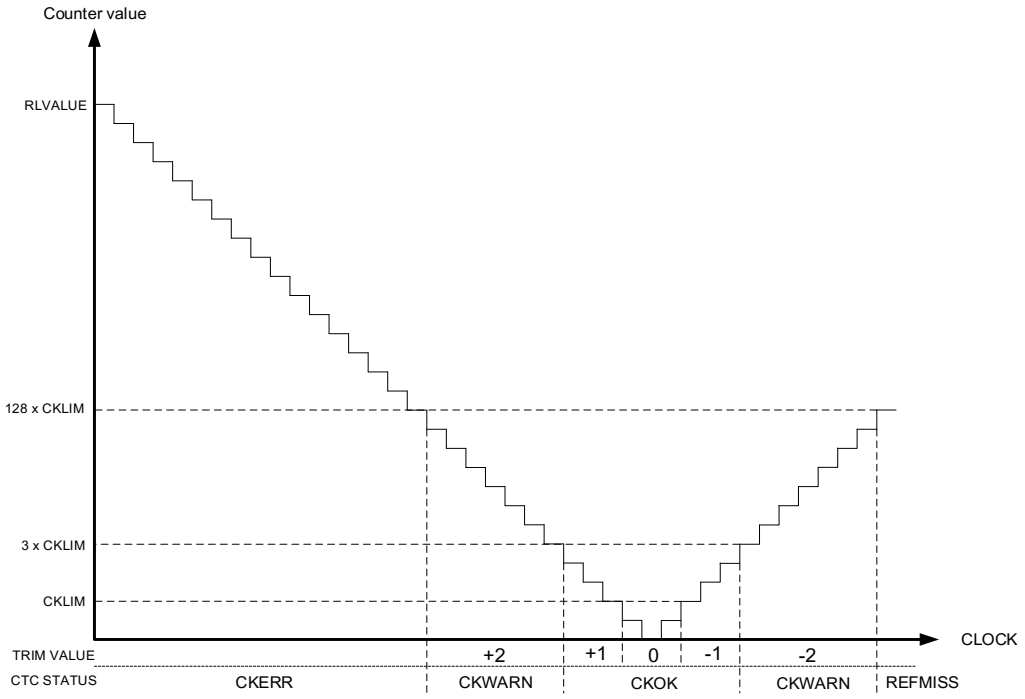
Thirdly, if a software reference pulse is needed, write 1 to SWREFPUL bit in CTC_CTL0 register. The software reference pulse generated in last step is logical OR with the external reference pulse.

6.3.2. CTC trim counter

The CTC trim counter is clocked by CK_IRC48M. After the CNTEN bit in CTC_CTL0 register is set, and a first REF sync pulse is detected, the counter starts down-counting from RLVALUE (defined in CTC_CTL1 register). If any REF sync pulse is detected, the counter reloads the RLVALUE and starts down-counting again. If no REF sync pulse is detected, the

counter down-counts to zero, and then up-counts to 128 x CKLIM (defined in CTC_CTL1 register), and then stops until next REF sync pulse is detected. If any REF sync pulse is 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 shows in ***Figure 6-2. CTC trim counter***.

Figure 6-2. CTC trim counter



6.3.3. Frequency evaluation and automatic trim process

The clock frequency evaluation is performed when a REF sync pulse occurs. 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 increase the 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 the TRIMVALUE in CTC_CTL0 register. The CKOKIF, CKWARNIF, CKERR and REFMISS in CTC_STAT register show the frequency evaluation scope.

If the AUTOTRIM bit in CTC_CTL0 register is set, the automatic hardware trim mode is 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 to raise the clock frequency automatically. Vice versa when it occurs on up-counting, the TRIMVALUE will be decreased to reduce the clock frequency automatically.

- Counter < CKLIM when REF sync pulse is detected.
  - When the CKOKIF in CTC_STAT register is set, an interrupt will be generated if CKOKIE bit in CTC_CTL0 register is 1.
  - If the AUTOTRIM bit in CTC_CTL0 register is set, the TRIMVALUE in CTC_CTL0

register is not changed.

- CKLIM <= Counter < 3 x CKLIM when REF sync pulse is detected.
  - When the CKOKIF in CTC_STAT register is set, an interrupt will be generated if CKOKIE bit in CTC_CTL0 register is 1.
  - If the AUTOTRIM bit in CTC_CTL0 register is set, the TRIMVALUE in CTC_CTL0 register adds 1 when down-counting or subtracts 1 when up-counting.
- 3 x CKLIM <= Counter < 128 x CKLIM when REF sync pulse is detected.
  - When the CKWARNIF in CTC_STAT register is set, an interrupt will be generated if CKWARNIE bit in CTC_CTL0 register is 1.
  - If the AUTOTRIM bit in CTC_CTL0 register is set, the TRIMVALUE in CTC_CTL0 register adds 2 when down-counting or subtracts 2 when up-counting.
- Counter >= 128 x CKLIM when down-counting and a REF sync pulse is detected.
  - When the CKERR in CTC_STAT register is set, an interrupt will be 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.
  - When the REFMISS in CTC_STAT register is set, an interrupt will be 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 0, the underflow will be occurred. The TRIMVALUE ranges from 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 will be generated if ERRIE bit in CTC_CTL0 register is 1.

6.3.4. Software program guide

The RLVALUE and CKLIM bits in CTC_CTL1 register are critical to evaluate the clock frequency and automatic 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 occurs when the CTC counter is zero, so the RLVALUE is:

$$RLVALUE=(F_{clock}/ F_{REF})-1 \quad (5\text{-}1)$$

The CKLIM is set by user according to the clock accuracy. It is recommend to set it to half of the step size, so the CKLIM is:

$$CKLIM=(F_{clock}/ F_{REF}) \times 0.12\% /2 \quad (5\text{-}2)$$

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.

6.4. Register definition

CTC base address: 0x4000 6C00

6.4.1. Control register 0 (CTC_CTL0)

Address offset: 0x00
Reset value: 0x0000 2000

This register 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		TRIMVALUE[5:0]						SWREFPUL	AUTOTRIM	CNTEN	Reserved	EREFIE	ERRIE	CKWARNIE	CKOKIE
		rw						w	rw	rw		rw	rw	rw	rw

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:8	TRIMVALUE[5:0]	IRC48M trim value When AUTOTRIM in CTC_CTL0 register is 0, these bits are set and cleared by software. This mode is used for software calibration. When AUTOTRIM in CTC_CTL0 register is 1, these bits are read only. The value is modified by hardware automatically. This mode is used for trim by hardware. The middle value is 32. When increasing 1, the IRC48M clock frequency adds around 57KHz. When decreasing 1, the IRC48M clock frequency subtracts around 57KHz.
7	SWREFPUL	Software reference source sync pulse This bit is set by software, and a reference sync pulse is generated to CTC counter. This bit is cleared by hardware automatically and read as 0. 0: No effect 1: Generates a software reference source sync pulse
6	AUTOTRIM	Hardware automatic trim mode This bit is set and cleared by software. When this bit is set, the hardware automatic trim is enabled, the TRIMVALUE bits in CTC_CTL0 register are modified by hardware automatically, until the frequency of IRC48M clock is closed to 48MHz. 0: Hardware automatic trim disabled 1: Hardware automatic trim enabled
5	CNTEN	CTC counter enable This bit is set and cleared by software. This bit is used to enable or disable the CTC trim counter. When this bit is set, the CTC_CTL1 register cannot be modified.

		0: CTC trim counter disabled 1: CTC trim counter enabled.
4	Reserved	Must be kept at reset value.
3	EREFIE	Expected reference (EREFIF) interrupt enable 0: EREFIF interrupt disable 1: EREFIF interrupt enable
2	ERRIE	Error (ERRIF) interrupt enable 0: ERRIF interrupt disable 1: ERRIF interrupt enable
1	CKWARNIE	Clock trim warning (CKWARNIF) interrupt enable 0: CKWARNIF interrupt disable 1: CKWARNIF interrupt enable
0	CKOKIE	Clock trim ok (CKOKIF) interrupt enable 0: CKOKIF interrupt disable 1: CKOKIF interrupt enable

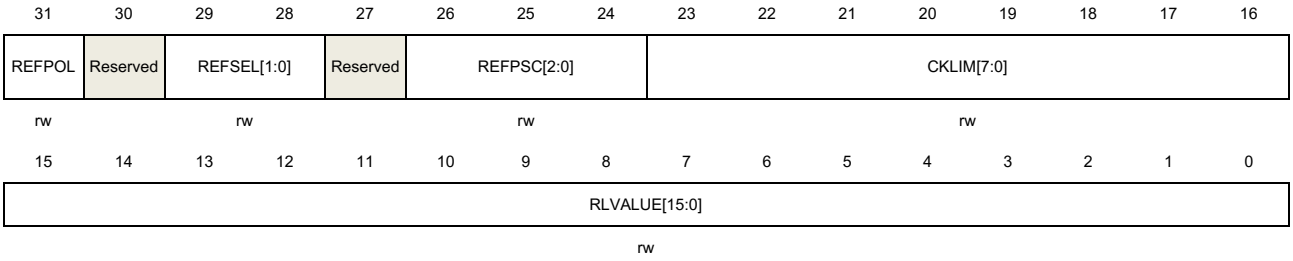
6.4.2. Control register 1 (CTC_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Note: This register cannot be modified when CNTEN is 1.



Bits	Fields	Descriptions
31	REFPOL	Reference signal source polarity This bit is set and cleared by software to select reference signal source polarity. 0: Rising edge selected 1: Falling edge selected
30	Reserved	Must be kept at reset value.
29:28	REFSEL[1:0]	Reference signal source selection These bits are set and cleared by software to select reference signal source. 00: GPIO (CTC_SYNC) selected 01: LXTAL clock selected

		10: Reserved 11: Reserved
27	Reserved	Must be kept at reset value.
26:24	REFPSC[2:0]	Reference signal source prescaler These bits are set and cleared by software. 000: Reference signal not divided 001: Reference signal divided by 2 010: Reference signal divided by 4 011: Reference signal divided by 8 100: Reference signal divided by 16 101: Reference signal divided by 32 110: Reference signal divided by 64 111: Reference signal divided by 128
23:16	CKLIM[7:0]	Clock trim base limit value These bits are set and cleared by software to define the clock trim base limit value. These bits are used for frequency evaluation and automatic trim process. Please refer to the ***Frequency evaluation and automatic trim process*** for detail.
15:0	RLVALUE[15:0]	CTC counter reload value 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 is received, so as to start or restart the counter.

6.4.3. Status register (CTC_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
REFCAP[15:0]															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFDIR	Reserved				TRIM ERR	REF MISS	CKERR	Reserved				EREFIF	ERRIF	CKWARN IF	CKOK IF
r					r	r	r					r	r	r	r

Bits	Fields	Descriptions
31:16	REFCAP[15:0]	CTC counter capture value. When a reference sync pulse occurs, the CTC trim counter value is captured to REFCAP bits.
15	REFDIR	CTC trim counter direction When a reference sync pulse occurs during the counter is working, the CTC trim

		counter direction is captured to REFDIR bit. 0: Up-counting 1: Down-counting
14:11	Reserved	Must be kept at reset value.
10	TRIMERR	Trim value error bit This bit is set by hardware when the TRIMVALUE in CTC_CTL0 register is overflow or underflow. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No trim value error occurs 1: Trim value error occurs
9	REFMISS	Reference sync pulse miss This bit is set by hardware when the reference sync pulse is missing. This occurs when the CTC trim counter reaches 128 x CKLIM during up-counting and no reference sync pulse is detected. This means the clock is too fast to be trimmed to the correct frequency or other error has occurred. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No Reference sync pulse miss occurs 1: Reference sync pulse miss occurs
8	CKERR	Clock trim error bit This bit is set by hardware when the clock trim error occurs. This occurs when the CTC trim counter is greater than or equal to 128 x CKLIM during down-counting when a reference sync pulse is detected. This means the clock is too slow and cannot be trimmed to the correct frequency. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No Clock trim error occurs 1: Clock trim error occurs
7:4	Reserved	Must be kept at reset value.
3	EREFIF	Expected reference interrupt flag This bit is set by hardware when the CTC counter reaches 0. When the EREFIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to EREFIC bit in CTC_INTC register. 0: No Expected reference occurs 1: Expected reference occurs
2	ERRIF	Error interrupt flag This bit is set by hardware when an error occurs. If any error of TRIMERR, REFMISS or CKERR occurs, this bit will be set. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register. 0: No Error occurs

1: An error occurs

1	CKWARNIF	Clock trim warning interrupt flag This bit is set by hardware when a clock trim warning occurs. If the CTC trim counter is greater than or equal to 3 x CKLIM and is smaller than 128 x CKLIM when a reference sync pulse is detected, this bit will be set. This means the clock is too slow or too fast, but can be trimmed to the correct frequency. The TRIMVALUE adds 2 or subtracts 2 when a clock trim warning occurs. When the CKWARNIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to CKWARNIC bit in CTC_INTC register. 0: No Clock trim warning occurs 1: Clock trim warning occurs
0	CKOKIF	Clock trim OK interrupt flag This bit is set by hardware when the clock trim is OK. If the CTC trim counter is smaller than 3 x CKLIM when a reference sync pulse is detected, this bit will be set. This means the clock is OK for using. The TRIMVALUE needs not to be adjusted. When the CKOKIE in CTC_CTL0 register is 1, an interrupt occurs. This bit is cleared by writing 1 to CKOKIC bit in CTC_INTC register. 0: No Clock trim OK occurs 1: Clock trim OK occurs

6.4.4. 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 This bit is written by software and read as 0. Write 1 to clear EREFIF bit in CTC_STAT register. Writing 0 has no effect.
2	ERRIC	ERRIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear ERRIF, TRIMERR, REFMISS and CKERR bits in CTC_STAT register. Writing 0 has no effect.

1	CKWARNIC	CKWARNIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear CKWARNIF bit in CTC_STAT register. Writing 0 has no effect.
0	CKOKIC	CKOKIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear CKOKIF bit in CTC_STAT register. Writing 0 has no effect.

7. Interrupt / event controller (EXTI)

7.1. Overview

Cortex®-M33 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and controls power management. It's tightly coupled to the processer core. More details about NVIC could be referred to Technical Reference Manual of Cortex®-M33.

EXTI (interrupt / event controller) contains up to 26 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.

7.2. Characteristics

- Cortex®-M33 system exception.
- Up to 104 maskable peripheral interrupts.
- 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 26 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

7.3. Function overview

The Arm Cortex®-M33 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. *Table 7-1. NVIC exception types in Cortex®-M33* and *Table 7-2. Interrupt vector table* list all exception types.

Table 7-1. NVIC exception types in Cortex®-M33

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

Table 7-2. Interrupt vector table

Interrupt number	Vector number	Peripheral interrupt description	Vector address
IRQ 0	16	WWDGT interrupt	0x0000_0040
IRQ 1	17	LVD from EXTI interrupt	0x0000_0044
IRQ 2	18	RTC Tamper and TimeStamp from EXTI 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 and CTC interrupts	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
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

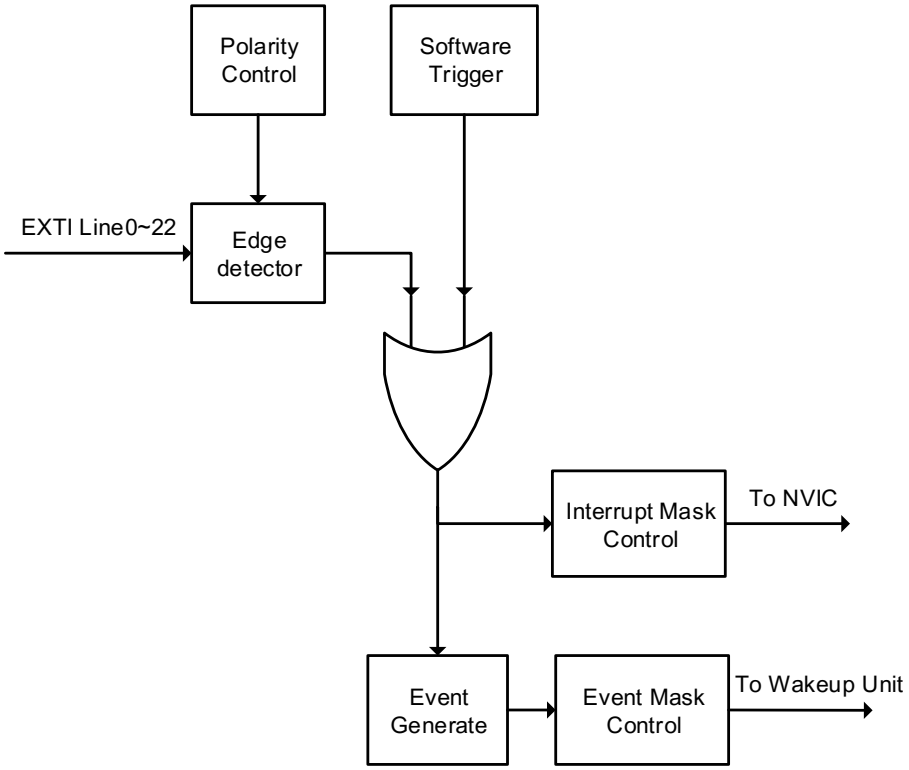
Interrupt number	Vector number	Peripheral interrupt description	Vector address
IRQ 18	34	ADC global interrupt	0x0000_0088
IRQ 19	35	CAN0 TX interrupt	0x0000_008C
IRQ 20	36	CAN0 RX0 interrupt	0x0000_0090
IRQ 21	37	CAN0 RX1 interrupt	0x0000_0094
IRQ 22	38	CAN0 EWMC interrupt	0x0000_0098
IRQ 23	39	EXTI line[9:5] interrupts	0x0000_009C
IRQ 24	40	TIMER0 break interrupt and TIMER8 global interrupts	0x0000_00A0
IRQ 25	41	TIMER0 update interrupt and TIMER9 global interrupts	0x0000_00A4
IRQ 26	42	TIMER0 trigger and Channel commutation interrupts and TIMER10 global interrupts	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 interrupt	0x0000_00D4
IRQ 38	54	USART1 global interrupt	0x0000_00D8
IRQ 39	55	USART2 global interrupt	0x0000_00DC
IRQ 40	56	EXTI line[15:10] interrupts	0x0000_00E0
IRQ 41	57	RTC alarm from EXTI interrupt	0x0000_00E4
IRQ 42	58	USBFS wakeup from EXTI interrupt	0x0000_00E8
IRQ 43	59	TIMER7 break interrupt and TIMER11 global interrupts	0x0000_00EC
IRQ 44	60	TIMER7 update interrupt and TIMER12 global interrupts	0x0000_00F0
IRQ 45	61	TIMER7 trigger and Channel commutation interrupts and TIMER13 global interrupts	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	SDIO global interrupt	0x0000_0104
IRQ 50	66	TIMER4 global interrupt	0x0000_0108

Interrupt number	Vector number	Peripheral interrupt description	Vector address
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 DAC0_OUT0, DAC0_OUT1 underrun error interrupts	0x0000_0118
IRQ 55	71	TIMER6 global interrupt	0x0000_011C
IRQ 56	72	DMA1 channel0 global interrupt	0x0000_0120
IRQ 57	73	DMA1 channel1 global interrupt	0x0000_0124
IRQ 58	74	DMA1 channel2 global interrupt	0x0000_0128
IRQ 59	75	DMA1 channel3 global interrupt	0x0000_012C
IRQ 60	76	DMA1 channel4 global interrupt	0x0000_0130
IRQ 61	77	Ethernet global interrupt	0x0000_0134
IRQ 62	78	Ethernet wakeup from EXTI interrupt	0x0000_0138
IRQ 63	79	CAN1 TX interrupts	0x0000_013C
IRQ 64	80	CAN1 RX0 interrupts	0x0000_0140
IRQ 65	81	CAN1 RX1 interrupt	0x0000_0144
IRQ 66	82	CAN1 EWMC interrupt	0x0000_0148
IRQ 67	83	USBFS global interrupt	0x0000_014C
IRQ 68	84	DMA1 channel5 global interrupt	0x0000_0150
IRQ 69	85	DMA1 channel6 global interrupt	0x0000_0154
IRQ 70	86	DMA1 channel7 global interrupt	0x0000_0158
IRQ 71	87	USART5 global interrupt	0x0000_015C
IRQ 72	88	I2C2 event interrupt	0x0000_0160
IRQ 73	89	I2C2 error interrupt	0x0000_0164
IRQ 74	90	USBHS endpoint 1 out interrupt	0x0000_0168
IRQ 75	91	USBHS endpoint 1 in interrupt	0x0000_016C
IRQ 76	92	USBHS wakeup from EXTI interrupt	0x0000_0170
IRQ 77	93	USBHS global interrupt	0x0000_0174
IRQ78	94	DCI global interrupt	0x0000_0178
IRQ79	95	Reserved	0x0000_017C
IRQ80	96	TRNG global interrupts	0x0000_0180
IRQ 81	97	FPU global interrupt	0x0000_0184
IRQ82	98	UART6 global interrupt	0x0000_0188
IRQ83	99	UART7 global interrupt	0x0000_018C
IRQ84	100	SPI3 global interrupt	0x0000_0190
IRQ85	101	SPI4 global interrupt	0x0000_0194
IRQ86	102	SPI5 global interrupt	0x0000_0198
IRQ87	103	SAI global interrupt	0x0000_019C
IRQ88	104	TLI global interrupt	0x0000_01A0
IRQ89	105	TLI global error interrupt	0x0000_01A4

Interrupt number	Vector number	Peripheral interrupt description	Vector address
IRQ90	106	IPA global interrupt	0x0000_01A8
IRQ91	107	PKCAU global interrupt	0x0000_01AC
IRQ92	108	I2C3 event interrupt	0x0000_01B0
IRQ93	109	I2C3 error interrupt	0x0000_01B4
IRQ94	110	I2C4 event interrupt	0x0000_01B8
IRQ95	111	I2C4 error interrupt	0x0000_01BC
IRQ96	112	I2C5 event interrupt	0x0000_01C0
IRQ97	113	I2C5 error interrupt	0x0000_01C4
IRQ98	114	I2C3 wakeup from EXTI interrupt	0x0000_01C8
IRQ99	115	I2C4 wakeup from EXTI interrupt	0x0000_01CC
IRQ100	116	I2C5 wakeup from EXTI interrupt	0x0000_01D0
IRQ101	117	SYSCFG ECC single bit error interrupt	0x0000_01D4
IRQ102	118	HAU global interrupt	0x0000_01D8
IRQ103	119	CAU global interrupt	0x0000_01DC

7.4. External interrupt and event block diagram

Figure 7-1. Block diagram of EXTI



7.5. External Interrupt and Event function overview

The EXTI contains up to 26 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 10 lines from internal modules which refers to ***Table 7-3. EXTI source*** for detail. 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*** section for detail).

EXTI can provide not only interrupts but also event signals to the processor. The Cortex®-M33 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 event. 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 PDx bits in EXTI_PD will be set when desired change is detected on these pins and thus, trigger interrupt or event for software. 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 PD bits will be set immediately and thus, trigger interrupts or events. Software should response to these interrupts, and clear related PDx bits.

Table 7-3. EXTI source

EXTI line number	Source
0	PA0 / PB0 / PC0 / PD0 / PE0 / PF0 / PG0 / PH0 / PI0

EXTI line number	Source
1	PA1 / PB1 / PC1 / PD1 / PE1 / PF1 / PG1 / PH1 / PI1
2	PA2 / PB2 / PC2 / PD2 / PE2 / PF2 / PG2 / PH2 / PI2
3	PA3 / PB3 / PC3 / PD3 / PE3 / PF3 / PG3 / PH3 / PI3
4	PA4 / PB4 / PC4 / PD4 / PE4 / PF4 / PG4 / PH4 / PI4
5	PA5 / PB5 / PC5 / PD5 / PE5 / PF5 / PG5 / PH5 / PI5
6	PA6 / PB6 / PC6 / PD6 / PE6 / PF6 / PG6 / PH6 / PI6
7	PA7 / PB7 / PC7 / PD7 / PE7 / PF7 / PG7 / PH7 / PI7
8	PA8 / PB8 / PC8 / PD8 / PE8 / PF8 / PG8 / PH8 / PI8
9	PA9 / PB9 / PC9 / PD9 / PE9 / PF9 / PG9 / PH9 / PI9
10	PA10 / PB10 / PC10 / PD10 / PE10 / PF10 / PG10 / PH10 / PI10
11	PA11 / PB11 / PC11 / PD11 / PE11 / PF11 / PG11 / PH11 / PI11
12	PA12 / 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	LVD
17	RTC Alarm
18	USBFS Wakeup
19	Ethernet Wakeup
20	USBHS Wakeup
21	RTC Tamper and TimeStamp event
22	RTC Wakeup
23	I2C3 Wakeup
24	I2C4 Wakeup
25	I2C5 Wakeup

7.6. Register definition

EXTI base address: 0x4001 3C00

7.6.1. Interrupt enable register (EXTI_INTEN)

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						INTEN25	INTEN24	INTEN23	INTEN22	INTEN21	INTEN20	INTEN19	INTEN18	INTEN17	INTEN16
						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
31:26	Reserved	Must be kept at reset value.
25:0	INTENx	Interrupt enable bit 0: Interrupt from Linex is disabled. 1: Interrupt from Linex is enabled.

7.6.2. Event enable register (EXTI_EVEN)

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						EVEN25	EVEN24	EVEN23	EVEN22	EVEN21	EVEN20	EVEN19	EVEN18	EVEN17	EVEN16
						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
31:26	Reserved	Must be kept at reset value.
25:0	EVENx	Event enable bit 0: Event from Linex is disabled. 1: Event from Linex is enabled.

7.6.3. Rising edge trigger enable register (EXTI_RTEN)

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						RTEN25	RTEN24	RTEN23	RTEN22	RTEN21	RTEN20	RTEN19	RTEN18	RTEN17	RTEN16
						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
31:26	Reserved	Must be kept at reset value.
25:0	RTENx	Rising edge trigger enable 0: Rising edge of Linex is invalid 1: Rising edge of Linex is valid as an interrupt / event request

7.6.4. Falling edge trigger enable register (EXTI_FTEN)

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						FTEN25	FTEN24	FTEN23	FTEN22	FTEN21	FTEN20	FTEN19	FTEN18	FTEN17	FTEN16
						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
31:26	Reserved	Must be kept at reset value.
25:0	FTENx	Falling edge trigger enable 0: Falling edge of Linex is invalid 1: Falling edge of Linex is valid as an interrupt / event request

7.6.5. Software interrupt event register (EXTI_SWIEV)

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						SWIEV25	SWIEV24	SWIEV23	SWIEV22	SWIEV21	SWIEV20	SWIEV19	SWIEV18	SWIEV17	SWIEV16
						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
31:26	Reserved	Must be kept at reset value.
25:0	SWIEVx	Interrupt/Event software trigger 0: Deactivate the EXTIx software interrupt / event request 1: Activate the EXTIx software interrupt / event request

7.6.6. Pending register (EXTI_PD)

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
Reserved						PD25	PD24	PD23	PD22	PD21	PD20	PD19	PD18	PD17	PD16
						rw	rw	rw	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
31:26	Reserved	Must be kept at reset value.
25:0	PDx	Interrupt pending status 0: EXTI Linex is not triggered 1: EXTI Linex is triggered. This bit is cleared to 0 by writing 1 to it.

8. General-purpose and alternate-function I/Os (GPIO and AFIO)

8.1. Overview

There are up to 140 general purpose I / O pins (GPIO), named PA0 ~ PA15, PB0 ~ PB15, PC0 ~ PC15, PD0 ~ PD15, PE0 ~ PE15, PF0 ~ PF15, PG0 ~ PG15, PH0 ~ PH15 and PI0 ~ PI11 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.

8.2. 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.

8.3. 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

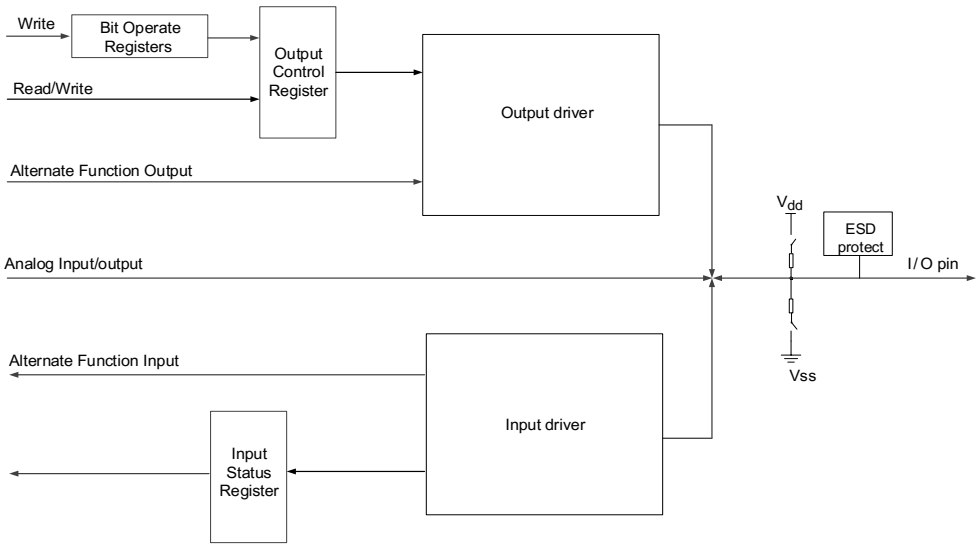
be configured by GPIO output speed registers (GPIOx_OSPD). Each port can be configured as floating (no pull-up and pull-down), pull-up or pull-down function by GPIO pull-up / pull-down registers (GPIOx_PUD).

Table 8-1. 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 8-1. Basic structure of a standard I / O shows the basic structure of an I / O port bit.

Figure 8-1. Basic structure of a standard I / O



8.3.1. **GPIO pin** configuration

During or just after the reset period, the alternative functions are all inactive and the GPIO ports are configured into the input floating 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 Floating 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.

8.3.2. **External interrupt / event lines**

The port can use external interrupt / event lines only if it is configured in input mode.

8.3.3. **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_AFSELz (z = 0,1)). The detail alternate function assignments for each port are in the device datasheet.

8.3.4. **Additional functions**

Some pins have additional functions, which have priority over the configuration in the standard GPIO registers. When for ADC or DAC 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.

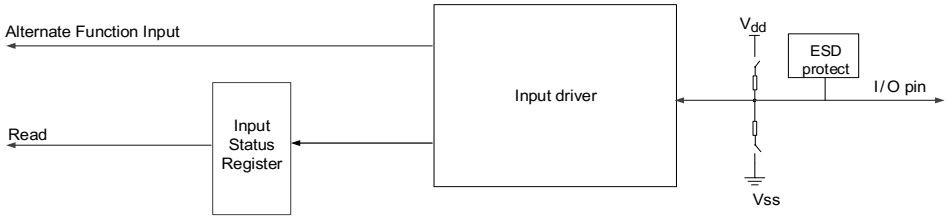
8.3.5. Input configuration

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.
- Disable the output buffer.

Figure 8-2. Basic structure of Input configuration shows the input configuration.

Figure 8-2. Basic structure of Input configuration



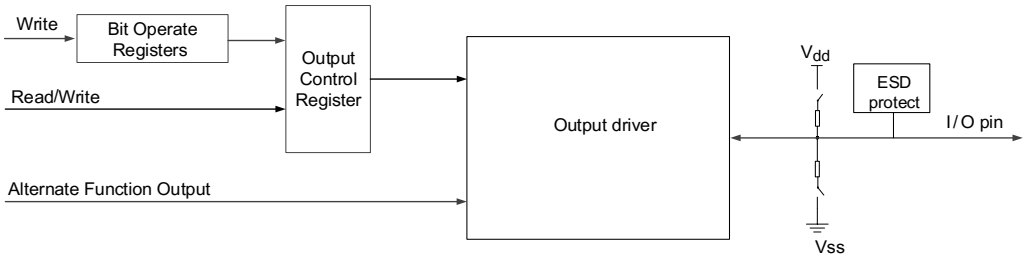
8.3.6. Output configuration

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 8-3. Basic structure of Output configuration shows the output configuration.

Figure 8-3. Basic structure of Output configuration



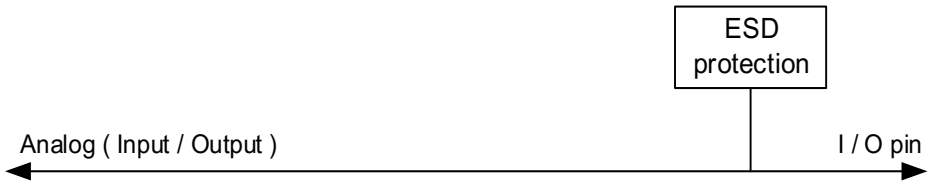
8.3.7. Analog configuration

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 8-4. Basic structure of Analog configuration shows the analog configuration.

Figure 8-4. Basic structure of Analog configuration



8.3.8. Alternate function (AF) configuration

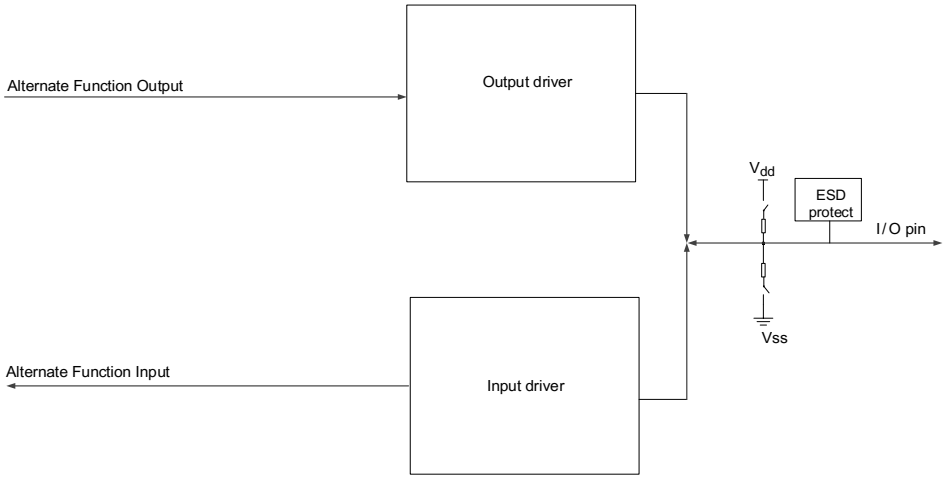
To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

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 8-5. Basic structure of Alternate function configuration shows the alternate function configuration.

Figure 8-5. Basic structure of Alternate function configuration



8.3.9. 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_AFSELz (z=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.

8.3.10. 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.

8.4. Register definition

GPIOA base address: 0x4002 0000

GPIOB base address: 0x4002 0400

GPIOC base address: 0x4002 0800

GPIOD base address: 0x4002 0C00

GPIOE base address: 0x4002 1000

GPIOF base address: 0x4002 1400

GPIOG base address: 0x4002 1800

GPIOH base address: 0x4002 1C00

GPIOI base address: 0x4002 2000

8.4.1. Port control register (GPIOx_CTL, x = A…I)

Address offset: 0x00

Reset value: 0xA800 0000 for port A; 0x0000 0280 for port B; 0x0000 0000 for others.

This register has to 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
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 These bits are set and cleared by software. Refer to CTL0[1:0] description
29:28	CTL14[1:0]	Pin 14 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
27:26	CTL13[1:0]	Pin 13 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
25:24	CTL12[1:0]	Pin 12 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description.

23:22	CTL11[1:0]	Pin 11 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
21:20	CTL10[1:0]	Pin 10 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
19:18	CTL9[1:0]	Pin 9 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
17:16	CTL8[1:0]	Pin 8 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
15:14	CTL7[1:0]	Pin 7 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
13:12	CTL6[1:0]	Pin 6 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
11:10	CTL5[1:0]	Pin 5 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
9:8	CTL4[1:0]	Pin 4 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
7:6	CTL3[1:0]	Pin 3 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
5:4	CTL2[1:0]	Pin 2 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
3:2	CTL1[1:0]	Pin 1 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
1:0	CTL0[1:0]	Pin 0 configuration bits These bits are set and cleared by software. 00: GPIO Input mode (reset value) 01: GPIO output mode 10: Alternate function mode

11: Analog mode (Input and Output)

8.4.2. Port output mode register (GPIOx_OMODE, x = A…I)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to 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
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 These bits are set and cleared by software. Refer to OM0 description
14	OM14	Pin 14 output mode bit These bits are set and cleared by software. Refer to OM0 description
13	OM13	Pin 13 output mode bit These bits are set and cleared by software. Refer to OM0 description
12	OM12	Pin 12 output mode bit These bits are set and cleared by software. Refer to OM0 description
11	OM11	Pin 11 output mode bit These bits are set and cleared by software. Refer to OM0 description
10	OM10	Pin 10 output mode bit These bits are set and cleared by software. Refer to OM0 description
9	OM9	Pin 9 output mode bit These bits are set and cleared by software. Refer to OM0 description
8	OM8	Pin 8 output mode bit These bits are set and cleared by software.

		Refer to OM0 description
7	OM7	Pin 7 output mode bit These bits are set and cleared by software. Refer to OM0 description
6	OM6	Pin 6 output mode bit These bits are set and cleared by software. Refer to OM0 description
5	OM5	Pin 5 output mode bit These bits are set and cleared by software. Refer to OM0 description
4	OM4	Pin 4 output mode bit These bits are set and cleared by software. Refer to OM0 description.
3	OM3	Pin 3 output mode bit These bits are set and cleared by software. Refer to OM0 description
2	OM2	Pin 2 output mode bit These bits are set and cleared by software. Refer to OM0 description
1	OM1	Pin 1 output mode bit These bits are set and cleared by software. Refer to OM0 description
0	OM0	Pin 0 output mode bit These bits are set and cleared by software. 0: Output push-pull mode (reset value) 1: Output open-drain mode

8.4.3. Port output speed register (GPIOx_OSPD, x = A…I)

Address offset: 0x08

Reset value: 0x0C00 0000 for port A; 0x0000 00C0 for port B; 0x0000 0000 for others.

This register has to 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
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 These bits are set and cleared by software. Refer to OSPD0[1:0] description
29:28	OSPD14[1:0]	Pin 14 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
27:26	OSPD13[1:0]	Pin 13 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
25:24	OSPD12[1:0]	Pin 12 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
23:22	OSPD11[1:0]	Pin 11 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description.
21:20	OSPD10[1:0]	Pin 10 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
19:18	OSPD9[1:0]	Pin 9 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
17:16	OSPD8[1:0]	Pin 8 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
15:14	OSPD7[1:0]	Pin 7 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
13:12	OSPD6[1:0]	Pin 6 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
11:10	OSPD5[1:0]	Pin 5 output max speed bits These bits are set and cleared by software. 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. Refer to OSPD0[1:0] description
7:6	OSPD3[1:0]	Pin 3 output max speed bits

		These bits are set and cleared by software. Refer to OSPD0[1:0] description
5:4	OSPD2[1:0]	Pin 2 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
3:2	OSPD1[1:0]	Pin 1 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
1:0	OSPD0[1:0]	Pin 0 output max speed bits These bits are set and cleared by software. 00: Output speed level 0 (reset state) 01: Output speed level 1 10: Output speed level 2 11: Output speed level 3

8.4.4. Port pull-up / pull-down register (GPIOx_PUD, x = A…I)

Address offset: 0x0C

Reset value: 0x6400 0000 for port A; 0x0000 0100 for port B; 0x0000 0000 for others.

This register has to 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
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 These bits are set and cleared by software. Refer to PUD0[1:0] description.
29:28	PUD14[1:0]	Pin 14 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
27:26	PUD13[1:0]	Pin 13 pull-up or pull-down bits 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 These bits are set and cleared by software. Refer to PUD0[1:0] description.

23:22	PUD11[1:0]	Pin 11 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
21:20	PUD10[1:0]	Pin 10 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
19:18	PUD9[1:0]	Pin 9 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
17:16	PUD8[1:0]	Pin 8 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
15:14	PUD7[1:0]	Pin 7 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
13:12	PUD6[1:0]	Pin 6 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
11:10	PUD5[1:0]	Pin 5 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
9:8	PUD4[1:0]	Pin 4 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
7:6	PUD3[1:0]	Pin 3 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
5:4	PUD2[1:0]	Pin 2 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description.
3:2	PUD1[1:0]	Pin 1 pull-up or pull-down bits These bits are set and cleared by software. 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

8.4.5. **Port input** status register (GPIOx_ISTAT, x = A…I)

Address offset: 0x10

Reset value: 0x0000 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															

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	Pin input status (y=0..15) These bits are set and cleared by hardware. 0: Input signal low 1: Input signal high

8.4.6. **Port output control register (GPIOx_OCTL, x = A…I)**

Address offset: 0x14

Reset value: 0x0000 0000

This register has to 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
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	Pin output control (y=0..15) These bits are set and cleared by software. 0: Pin output low 1: Pin output high

8.4.7. Port bit operate register (GPIOx_BOP, x = A…I)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to 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
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	Pin Clear bit y(y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLy bit 1: Clear the corresponding OCTLy bit to 0
15:0	BOPy	Pin Set bit y(y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLy bit 1: Set the corresponding OCTLy bit to 1

8.4.8. Port configuration lock register (GPIOx_LOCK, x = A…I)

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 sequence key It can only be setted using the Lock Key Writing Sequence. And can always be read. 0: GPIO_LOCK register is not locked and the port configuration is not locked. 1: GPIO_LOCK register is locked until an MCU reset. LOCK key configuration sequence. Write 1->Write 0->Write 1-> Read 0-> Read 1 **Note:** The value of LK[15:0] must hold during the LOCK Key Writing sequence.
15:0	LKy	Port Lock bit y (y=0..15) These bits are set and cleared by software. 0: The corresponding bit port configuration is not locked 1: The corresponding bit port configuration is locked when LKK bit is "1"

8.4.9. Alternate function selected register 0 (GPIOx_AFSEL0, x = A…I)

Address offset: 0x20

Reset value: 0x0000 0000

This register has to 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
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 These bits are set and cleared by software. Refer to SEL0 [3:0] description.
27:24	SEL6[3:0]	Pin 6 alternate function selected These bits are set and cleared by software. Refer to SEL0 [3:0] description.
23:20	SEL5[3:0]	Pin 5 alternate function selected These bits are set and cleared by software. Refer to SEL0 [3:0] description.
19:16	SEL4[3:0]	Pin 4 alternate function selected These bits are set and cleared by software. Refer to SEL0 [3:0] description.
15:12	SEL3[3:0]	Pin 3 alternate function selected These bits are set and cleared by software. Refer to SEL0 [3:0] description.

11:8	SEL2[3:0]	Pin 2 alternate function selected These bits are set and cleared by software. Refer to SEL0 [3:0] description.
7:4	SEL1[3:0]	Pin 1 alternate function selected These bits are set and cleared by software. Refer to SEL0 [3:0] description.
3:0	SEL0[3:0]	Pin 0 alternate function selected These bits are set and cleared by software. 0000: AF0 selected (reset value) 0001: AF1 selected 0010: AF2 selected 0011: AF3 selected … 1111: AF15 selected

8.4.10. Alternate function selected register 1 (GPIOx_AFSEL1, x = A…I)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to 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
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 These bits are set and cleared by software. Refer to SEL8[3:0] description.
27:24	SEL14[3:0]	Pin 14 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description.
23:20	SEL13[3:0]	Pin 13 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description.
19:16	SEL12[3:0]	Pin 12 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description.

15:12	SEL11[3:0]	Pin 11 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description.
11:8	SEL10[3:0]	Pin 10 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description.
7:4	SEL9[3:0]	Pin 9 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description.
3:0	SEL8[3:0]	Pin 8 alternate function selected These bits are set and cleared by software. 0000: AF0 selected (reset value) 0001: AF1 selected 0010: AF2 selected 0011: AF3 selected … 1111: AF15 selected

8.4.11. Bit clear register (GPIOx_BC, x = A…I)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to 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
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	Pin Clear bit y (y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLy bit 1: Clear the corresponding OCTLy bit

8.4.12. Port bit toggle register (GPIOx_TG, x = A…I)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to 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
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	Pin toggle bit y (y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLy bit 1: Toggle the corresponding OCTLy bit

9. Cyclic redundancy checks management unit (CRC)

9.1. 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.

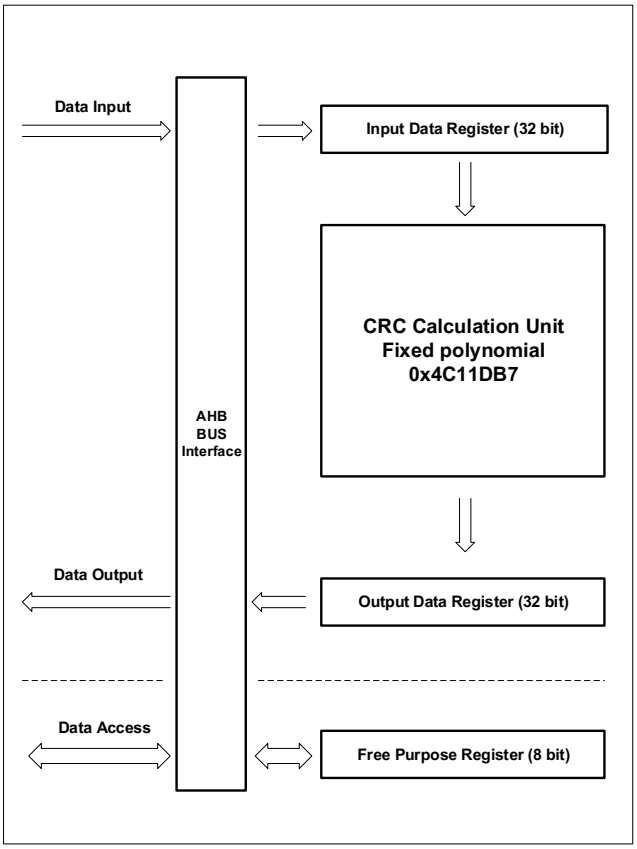
This CRC calculation unit can be used to calculate 32 bit CRC code with fixed polynomial.

9.2. Characteristics

- 32-bit data input and 32-bit data output. Calculation period is 4 AHB clock cycles for 32-bit input data size from data entered to the calculation result available.
- Free 8-bit register is unrelated to calculation and can be used for any other goals by any other peripheral devices.
- Fixed polynomial: 0x4C11DB7
  $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$

This 32-bit CRC polynomial is a common polynomial used in Ethernet.

Figure 9-1. Block diagram of CRC calculation unit



9.3. Function overview

- CRC management 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 software setting the CRC_CTL register, the new input raw data will be calculated based on the result of previous value of CRC_DATA.

  During CRC calculation 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, any value you write in will be read out at anytime.

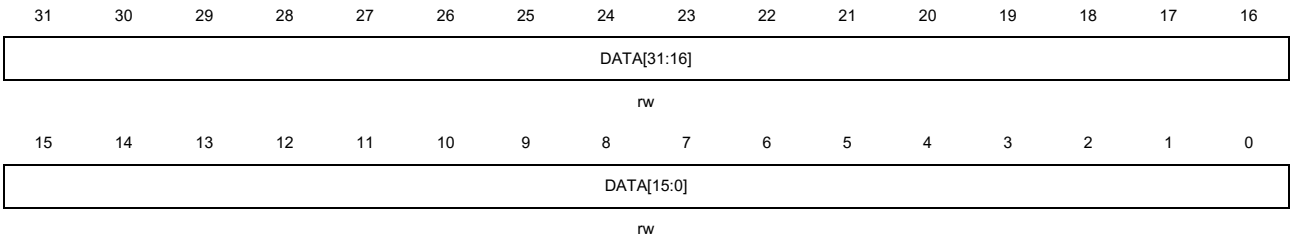
9.4. Register definition

CRC base address: 0x4002 3000

9.4.1. Data register (CRC_DATA)

Address offset: 0x00
Reset value: 0xFFFF FFFF

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

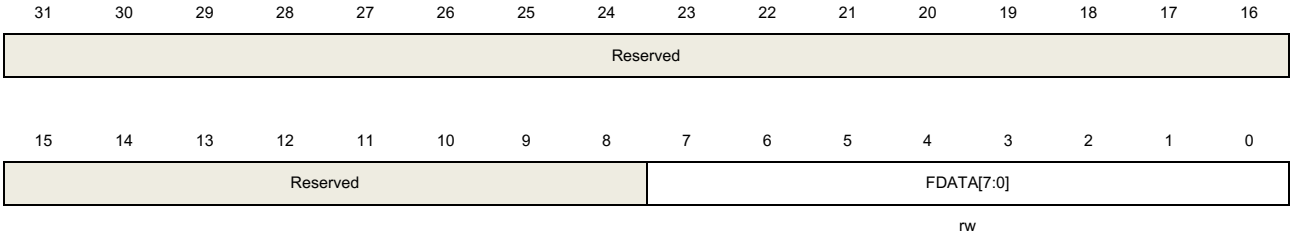


Bits	Fields	Descriptions
31:0	DATA[31:0]	CRC calculation result bits Software writes and reads. This register is used to calculate new data, and the register can be written the new data directly. Written value cannot be read because the read value is the previous CRC calculation result.

9.4.2. Free data register (CRC_FDATA)

Address offset: 0x04
Reset value: 0x0000 0000

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



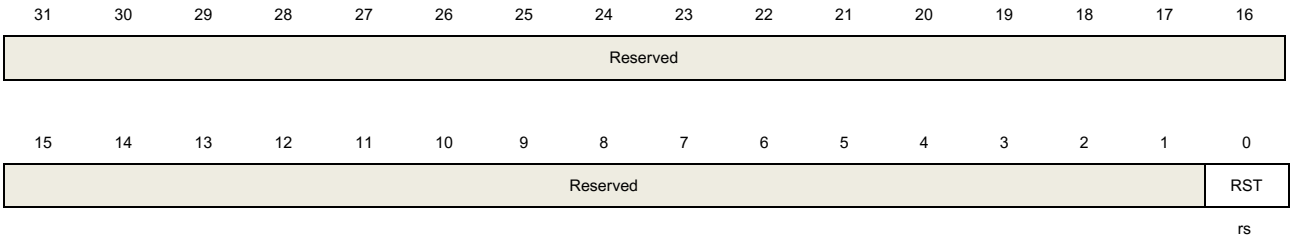
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	FDATA[7:0]	Free Data Register bits Software writes and reads. These bits are unrelated with CRC calculation. This byte can be used for any goal by any other peripheral. The CRC_CTL register will take no effect to the byte.

9.4.3. Control register (CRC_CTL)

Address offset: 0x08

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	RST	Set this bit can reset the CRC_DATA register to the value of 0xFFFFFFFF then automatically cleared itself to 0 by hardware. This bit will take no effect to CRC_FDATA. Software writes and reads.

10. True random number generator (TRNG)

10.1. Overview

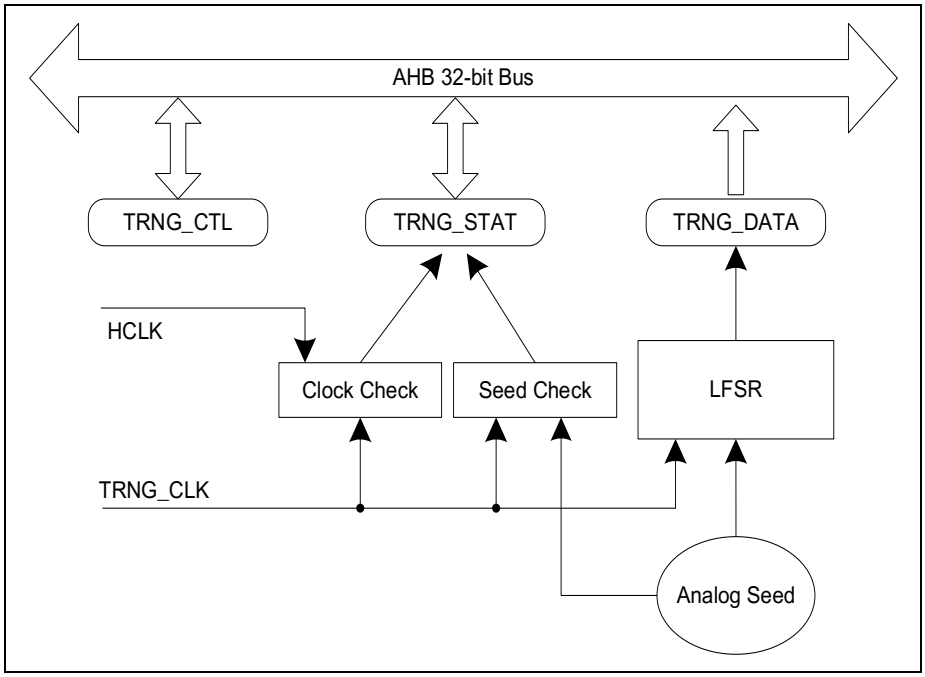
The true random number generator (TRNG) module can generate a 32-bit random value by using continuous analog noise.

10.2. Characteristics

- About 40 periods of TRNG_CLK are needed between two consecutive random numbers
- 32-bit random value seed is generated from analog noise, so the random number is a true random number.

10.3. Function overview

Figure 10-1. TRNG block diagram



The random number seed comes from analog circuit. This analog seed is then plugged into a linear feedback shift register (LFSR), where a 32-bit width random number is generated.

The analog seed is generated by several ring oscillators. The LFSR is driven by a configurable TRNG_CLK (refer to ***Reset and clock unit (RCU)*** chapter), so that the quality of the generated random number depends on TRNG_CLK exclusively, no matter what HCLK frequency was set or not.

The 32-bit value of LFSR will be transferred into TRNG_DATA register after a sufficient

number of seeds have been sent to the LFSR.

At the same time, the analog seed and TRNG_CLK clock are monitored. When an analog seed error or a clock error occurs, the corresponding status bit in TRNG_STAT will be set and an interrupt will generate if the TRNGIE bit in TRNG_CTL is set.

10.3.1. Operation flow

The following steps are recommended for using TRNG block:

1). Enable the interrupt as necessary, so that when a random number or an error occurs, an interrupt will be generated.

2). Enable the TRNGEN bit.

3). When an interrupt occurs, check the status register TRGN_STAT, if SEIF=0, CEIF=0 and DRDY=1, then the random value in the data register could be read.

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.

10.3.2. Error flags

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.

Seed Error

When the analog seed is not changed or always changing during 64 TRNG_CLK periods, the SECS and SEIF bit will be set. In this case, the random data in data register should not be used. The application needs to clear the SEIF bit, then clear and set TRNGEN bit for restarting the TRNG.

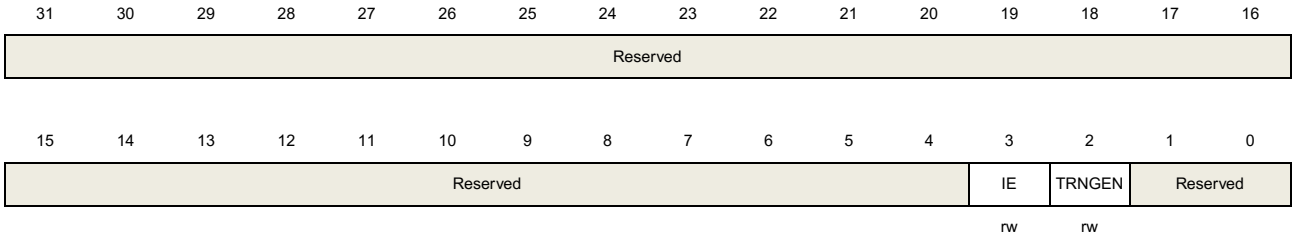
10.4. Register definition

TRNG base address: 0x5006 0800

10.4.1. Control register (TRNG_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).

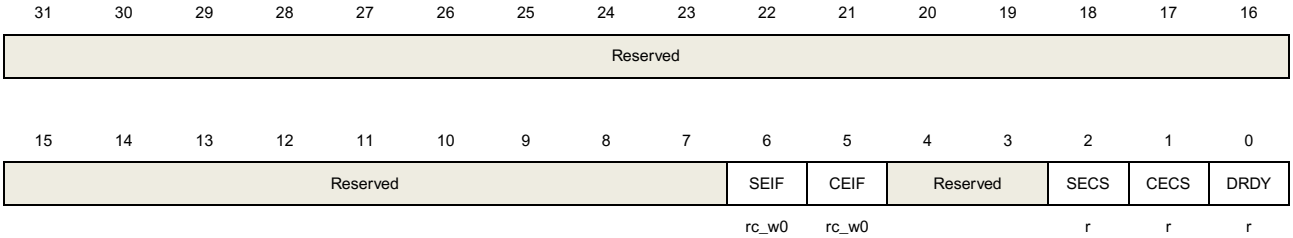


Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3	TRNGIE	Interrupt enabled bit. This bit controls the generation of an interrupt when DRDY,SEIF or CEIF was set 0: Disable TRNG Interrupt 1: Enable TRNG Interrupt
2	TRNGEN	TRNG enabled bit. 0: Disable TRNG module 1: Enable TRNG module
1:0	Reserved	Must be kept at reset value.

10.4.2. Status register (TRNG_STAT)

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
31:7	Reserved	Must be kept at reset value.
6	SEIF	Seed error interrupt flag This bit will be set if more than 64 consecutive same bit or more than 32 consecutive 01(or 10) changing are detected. 0: No fault detected 1: Seed error has been detected. The bit is cleared by writing 0.
5	CEIF	Clock error interrupt flag This bit will be set if TRNG_CLK frequency is lower than 1/16 HCLK frequency. 0: No fault detected 1: Clock error has been detected. The bit is cleared by writing 0.
4:3	Reserved	Must be kept at reset value
2	SECS	Seed error current status 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. 1: Seed error is detected at current time if more than 64 consecutive same bits or more than 32 consecutive 01(or 10) changing are detected
1	CECS	Clock error current status 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. 0: The content of TRNG data register is not available. 1: The content of TRNG data register is available

10.4.3. Data register (TRNG_DATA)

Address offset: 0x08

Reset value: 0x0000 0000

Application must make sure DRDY is set before reading 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
TRNGDATA[31:16]															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNGDATA[15:0]															
r															

Bits	Fields	Descriptions
31:0	TRNGDATA[31:0]	32-Bit Random data

11. Public Key Cryptographic Acceleration Unit (PKCAU)

11.1. Overview

Public key encryption is also called asymmetric encryption, asymmetric encryption algorithms use different keys for encryption and decryption. The Public Key Cryptographic Acceleration Unit (PKCAU) can accelerate RSA (Rivest, Shamir and Adleman), Diffie-Hellmann (DH key exchange) and ECC (elliptic curve cryptography) in GF(p) (Galois domain). These operations are performed in the Montgomery domain to improve computational efficiency.

11.2. Characteristics

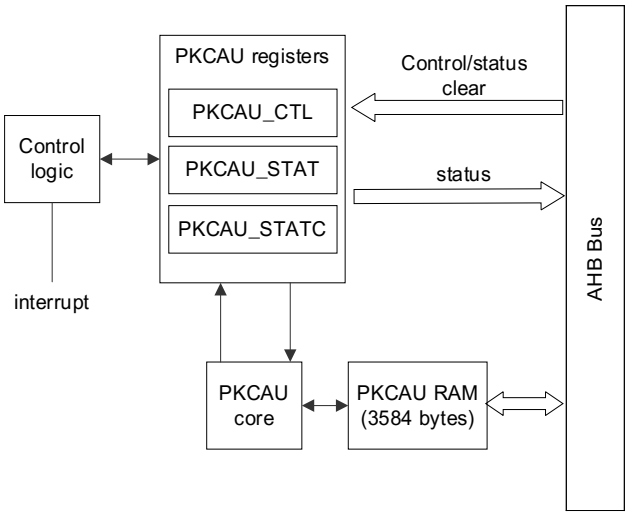
- Support RSA / DH algorithms with up to 3136 bits of operands.
- Support ECC algorithm with up to 640 bits of operands.
- RSA modular exponentiation, RSA CRT exponentiation.
- ECC scalar multiplication, check point on elliptic curve.
- ECDSA (Elliptic Curve Digital Signature Algorithm) signature and verification.
- Support Montgomery multiplication, accelerate RSA, DH and ECC operations.
- Embedded RAM of 3584 bytes.
- Conversion between the Montgomery domain and the natural domain.
- PKCAU is a 32 bit peripheral, only 32-bit access is supported.

11.3. Function overview

The Public Key Cryptographic Acceleration Unit (PKCAU) is used for accelerating RSA, DH and ECC operations in GF(p) domain. The PKCAU module contains PKCAU RAM, PKCAU core, and PKCAU registers. The PKCAU RAM is used to store the parameters required by the operation and save the calculation result after the calculation is completed.

Figure 11-1. PKCAU module block diagram below provides details on the internal configuration of the PKCAU interface.

Figure 11-1. PKCAU module block diagram



11.3.1. Operands

If the RSA operand size is ROS, the modulus length is ML, then the data size is ROS = (ML/32+1) words. If the ECC operand size is EOS, the prime modulus length is ML, then the data size is EOS = (ML / 32 + 1) words.

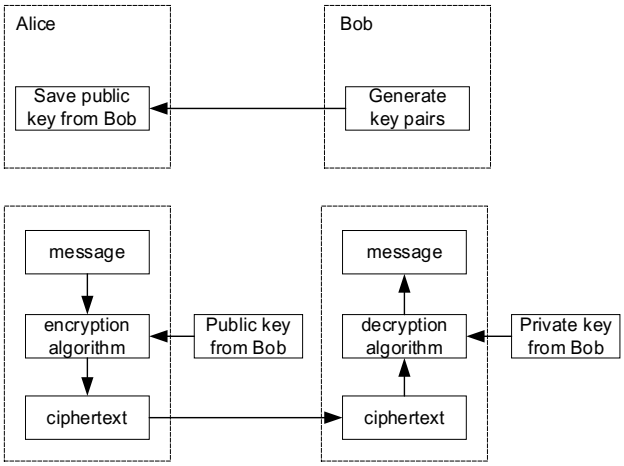
The PKCAU supports RSA / DH algorithms with up to 3136 bits (98 words) of operands and ECC algorithm with up to 640 bits (20 words) of operands. The maximum ROS is 99 words, and the maximum EOS is 21 words.

When writing the input parameters to the PKCAU RAM, a word 0x00000000 must be added. The PKCAU RAM is little-endian. For example, when writing the input parameter x_p of ECC P256 for ECC scalar multiplication to PKCAU RAM, the modulus length is 8 words, address offset 0x55C stores the lowest byte, address offset 0x578 stores the highest byte. Address offset 0x57C stores word 0x00000000.

11.3.2. RSA algorithm

RSA algorithm is a common public key cryptography algorithm and the most widely used asymmetric cryptography algorithm. The RSA algorithm flow is shown in ***Figure 11-2. Flow chart of RSA algorithm***.

Figure 11-2. Flow chart of RSA algorithm



A complete public key crypto system includes key pairs (public and private keys), encryption algorithms and decryption algorithms.

Generate key pairs of RSA

1. Select two large primes p and q (p != q);
2. Calculate n=p×q, n is the modulus of public and private keys;
3. Calculate L = Ø (n) = (p - 1)(q - 1), Ø (n) is euler function;
4. Select e, 1<e<L, e and L must be relatively prime;
5. Calculate d, 1<d<L and e * d mod L = 1.

The parameters show in ***Table 11-1. Parameters of RSA algorithm*** can be obtained by the above calculation.

Table 11-1. Parameters of RSA algorithm

Parameters	Description
n	modulus
e	public exponent
d	private exponent
(n,e)	public key
(n,d)	private key

RSA encryption

Bob generates a key pair that conforms to RSA algorithm standard, including public key and private key. Bob sends the public key to Alice, and holds the private key. Alice can encrypt the message m through the public key from Bob and obtain the ciphertext c. Then Alice sends the ciphertext to Bob. The ciphertext is $c = m^e \bmod n$.

RSA decryption

After receiving the ciphertext, Bob decrypts the ciphertext to get the plaintext by the private key. The decryption process is $m = c^d \bmod n$.

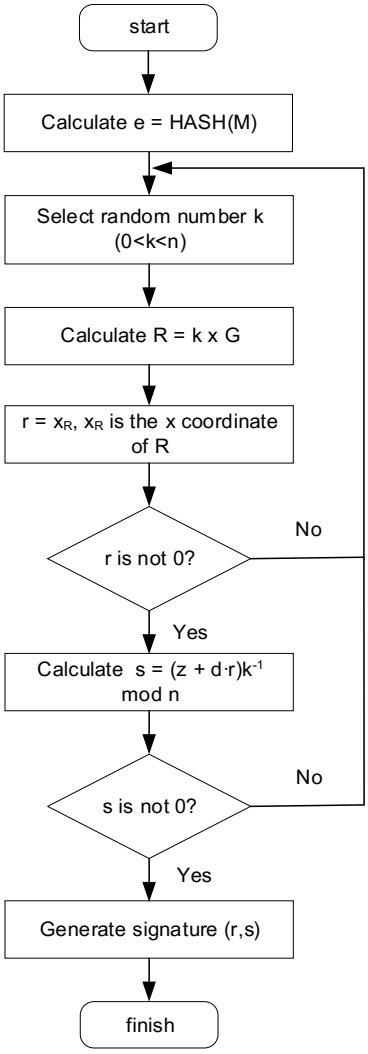
11.3.3. ECC algorithm

Suppose the message is M, d is the private key, G is the base point of the chosen elliptic curve, Q is a point of the chosen elliptic curve, with a prime order n. The hash function is HASH(), z is the Ln leftmost bits of HASH (M), where Ln is the bit length of the order n. The ECDSA sign and verification are detailed as follows:

ECDSA sign

The signature result of ECDSA consists of r and s. The process to generate ECDSA signature is shown in *Figure 11-3. Flow chart of ECDSA sign*.

Figure 11-3. Flow chart of ECDSA sign

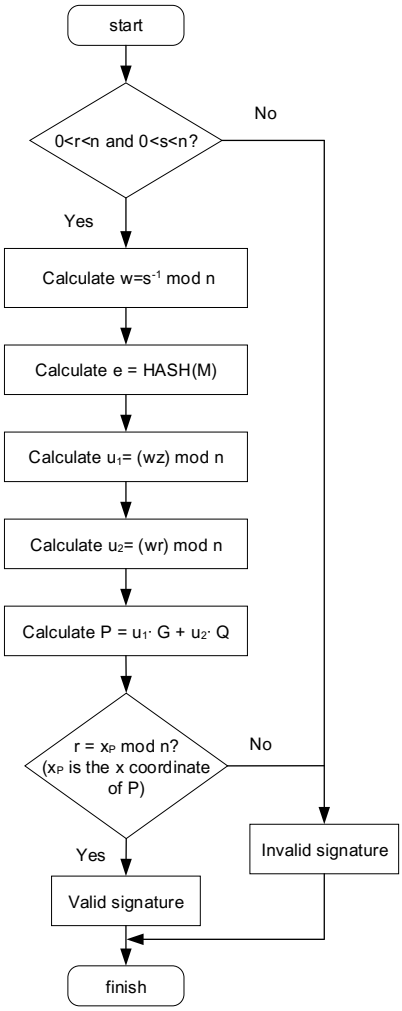


ECDSA verification

Before verifying the signature, be sure to get the signer's public key, message, and signature. The process to generate ECDSA signature is shown in *Figure 11-4. Flow chart of ECDSA*

verification.

Figure 11-4. Flow chart of ECDSA verification



Note: The HASH in the above diagram is the agreed cryptographic hash function.

11.3.4. Integer arithmetic operations

The integer arithmetic operation can be selected by configuring the MODSEL[5:0] in PKCAU_CTL register. The operation modes to be selected is shown in *Table 11-2. Integer arithmetic operations*.

Table 11-2. Integer arithmetic operations

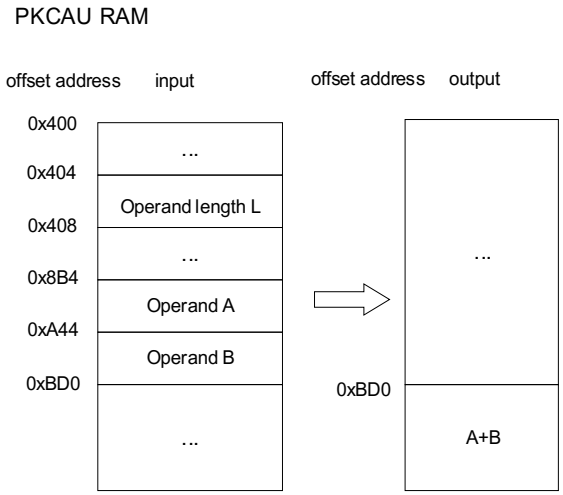
MODSEL[5:0]	Operation modes
000000	calculate Montgomery parameter and then modular exponentiation
000001	calculate Montgomery parameter only
000010	modular exponentiation (the Montgomery parameter must be preloaded)
000111	RSA CRT exponentiation
001000	Modular inversion

MODSEL[5:0]	Operation modes
001001	Arithmetic addition
001010	Arithmetic subtraction
001011	Arithmetic multiplication
001100	Arithmetic comparison
001101	Modular reduction
001110	Modular addition
001111	Modular subtraction
010000	Montgomery multiplication

Arithmetic addition

The arithmetic addition operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001001". The operation declaration is shown in ***Figure 11-5. Arithmetic addition***. The operation result is "result = A + B".

Figure 11-5. Arithmetic addition



$0<=A<2^{L}$, $0<=B<2^{L}$, $0<=result<2^{L+1}$, $0<L<=3136$.

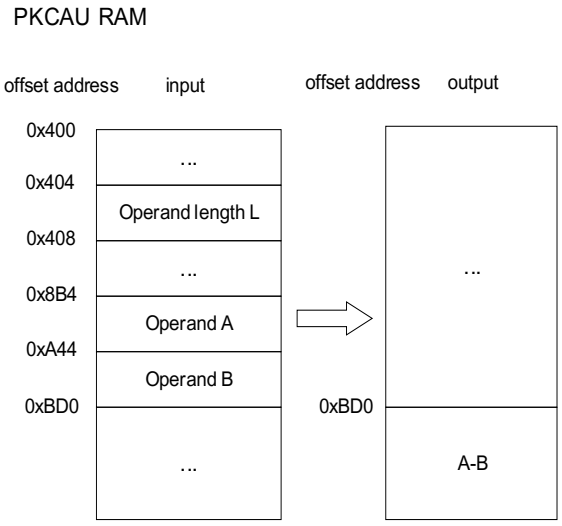
Arithmetic subtraction

The arithmetic subtraction operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001010". The operation declaration is shown in ***Figure 11-6. Arithmetic subtraction***.

If A>=B, the operation result is "result = A-B".

If A<B, the operation result is "result = $A-B+2^{L+ceil(L\%32)}$".

Figure 11-6. Arithmetic subtraction

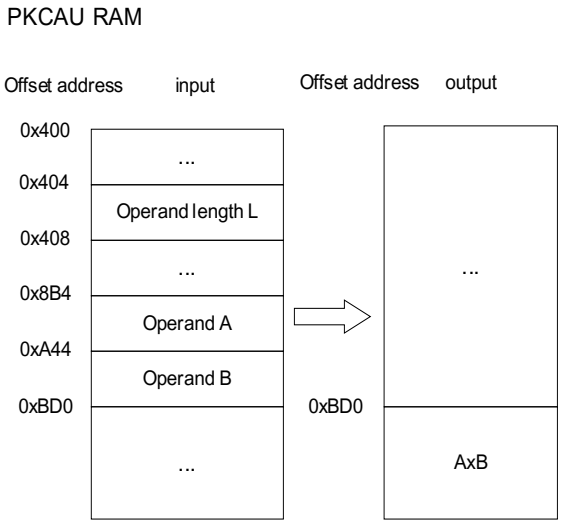


$0<=A<2^L$, $0<=B<2^L$, $0<=result<2^L$, $0<L<=3136$.

Arithmetic multiplication

The arithmetic multiplication operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001011". The operation declaration is shown in ***Figure 11-7. Arithmetic multiplication***. The operation result is "result = A × B".

Figure 11-7. Arithmetic multiplication



$0<=A<2^L$, $0<=B<2^L$, $0<=result<2^{2L}$, $0<L<=3136$.

Arithmetic comparison

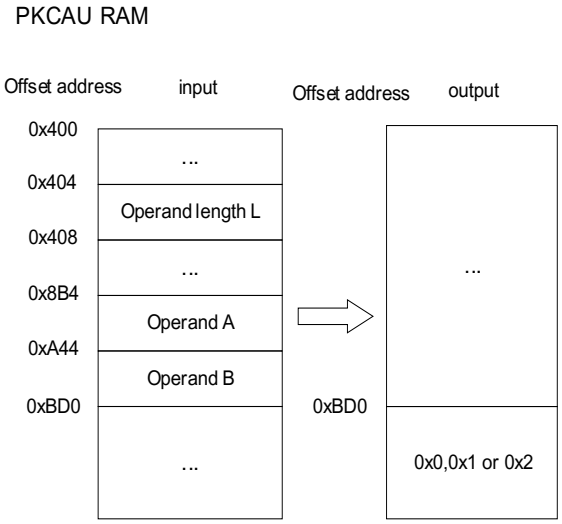
The arithmetic comparison operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001100". The operation declaration is shown in ***Figure 11-8. Arithmetic comparison***.

If A=B, the operation result is "result = 0x0";

If A>B, the operation result is "result = 0x1";

If A<B, the operation result is "result = 0x2".

Figure 11-8. Arithmetic comparison

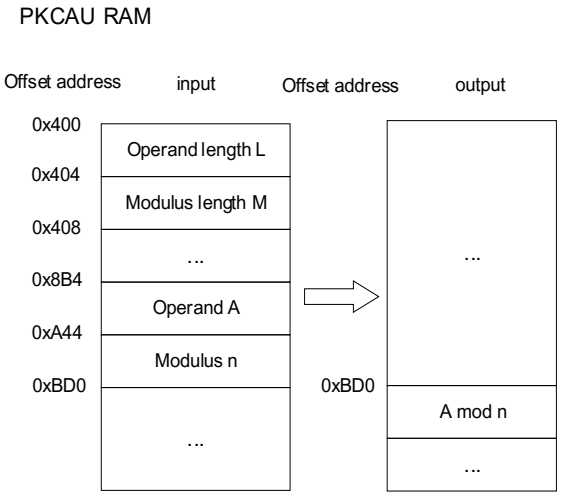


$0<=A<2^L$, $0<=B<2^L$, result = 0x0, 0x01, 0x2, $0<L<=3136$.

Modular reduction

The modular reduction operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001101". The operation declaration is shown in ***Figure 11-9. Modular reduction***. The operation result is "result = A mod n".

Figure 11-9. Modular reduction

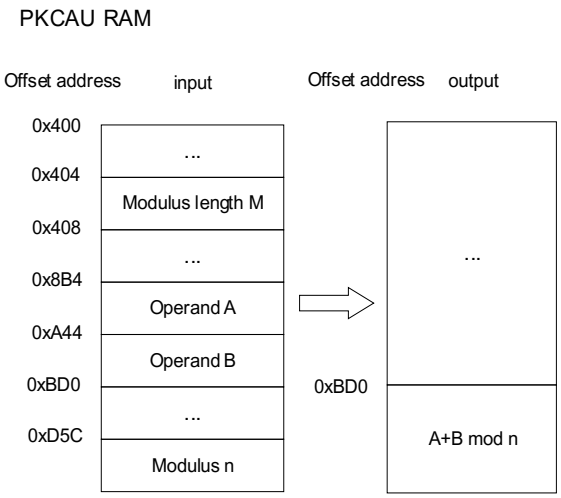


$0<L<=3136$, $0<M<=3136$, $0<=A<2^L$, $0<n<2^M$, $0<=result<n$.

Modular addition

The modular addition operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001110". The operation declaration is shown in *Figure 11-10. Modular addition*. The operation result is "result = A+B mod n".

Figure 11-10. Modular addition



$0<=A<n$, $0<=B<n$, $0<=result<n$, $0<n<2^M$, $0<M<=3136$.

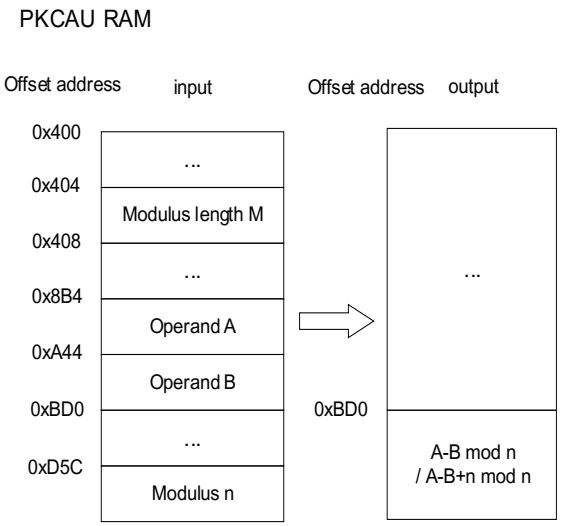
Modular subtraction

The modular subtraction operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001111". The operation declaration is shown in *Figure 11-11. Modular subtraction*.

If A>=B, the operation result is "result = A-B mod n";

If A<B, the operation result is "result = A-B+n mod n".

Figure 11-11. Modular subtraction



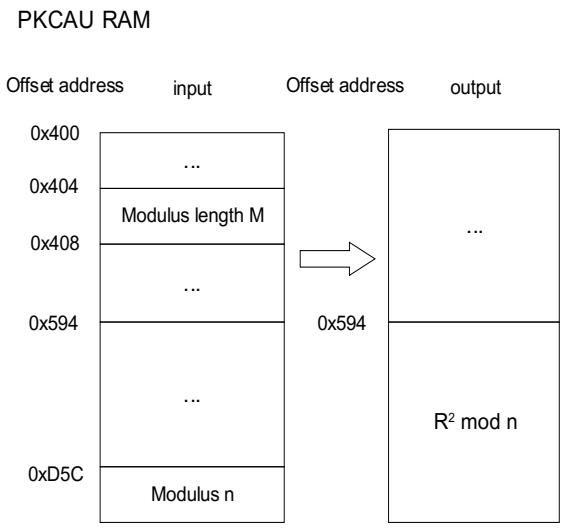
0<=A<n, 0<=B<n, 0<=result<n, 0<n<2^M, 0<M<=3136.

Montgomery parameter calculation

PKCAU needs to use the Montgomery parameter (R^2 mod n) to convert the operands to Montgomery residue system representation.

The Montgomery parameter calculation operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "000001". The operation declaration is shown in ***Figure 11-12. Montgomery parameter calculation***.

Figure 11-12. Montgomery parameter calculation



0<M<=3136, 1<n<2^M(n must be odd integer).

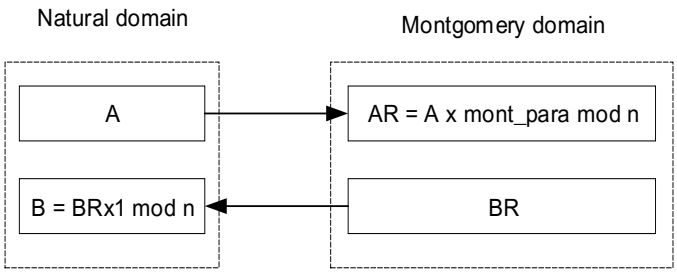
Montgomery multiplication

Suppose A, B and C are in natural domain. "x" function is Montgomery multiplication operation. The two main uses of this opreation are as follows:

1. Mutual mapping between Montgomery domain and natural domain.

As is shown in ***Figure 11-13. Mutual mapping between Montgomery domain and natural domain***, if A is an integer in natural domain, the Montgomery parameter mont_para is R^2 mod n, the result AR = A x mont_para mod n is A in Montgomery domain. In turn, if BR is an integer in Montgomery domain, the calculation result B = BR x 1 mod n is in natural domain.

Figure 11-13. Mutual mapping between Montgomery domain and natural domain



2. Perform a modular multiplication operation A x B mod n.

(1) Calculate Montgomery parameter mont_para = R^2 mod n.

(2) Calculate AR = A x mont_para mod n, the output is in Montgomery domain.

(3) Calculate AB = AR x B mod n, the output is in natural domain.

Multiple modular multiplication A x B x C mod n.

(1) Calculate Montgomery parameter mont_para = R^2 mod n.

(2) Calculate AR = A x mont_para mod n, the output is in Montgomery domain.

(3) Calculate BR = B x mont_para mod n, the output is in Montgomery domain.

(4) Calculate ABR = AR x BR mod n, the output is in Montgomery domain.

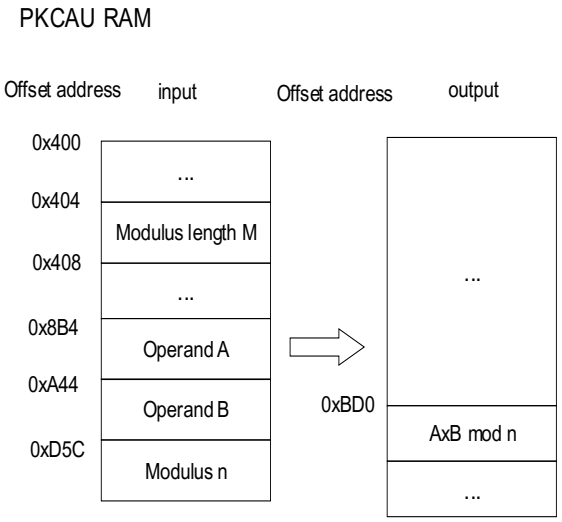
(5) Calculate CR = C x mont_para mod n, the output is in Montgomery domain.

(6) Calculate ABCR = ABR x CR mod n, the output is in Montgomery domain.

(7) Calculate ABC= ABCR x 1 mod n, the output is in natural domain.

The Montgomery multiplication operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "010000". The operation declaration is shown in ***Figure 11-14. Montgomery multiplication***.

Figure 11-14. Montgomery multiplication



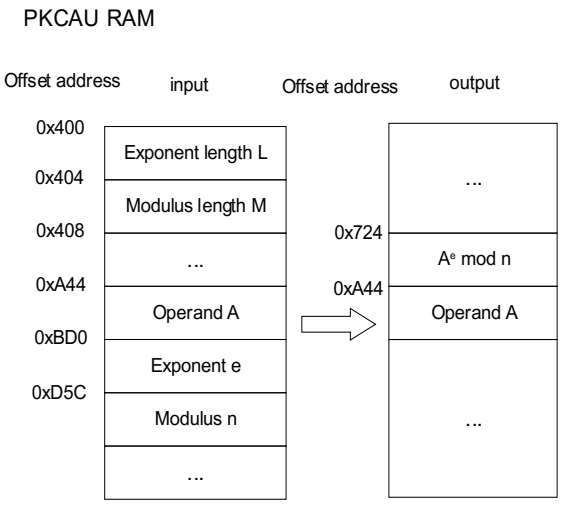
0<=A<n, 0<=B<n, 0<n<2^M, 0<M<=3136 (n must be odd integer).

Modular exponentiation

Normal mode

The Modular exponentiation of normal mode operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "000000". The operation declaration is shown in ***Figure 11-15. Modular exponentiation of normal mode***. The operation result is "result = A^e mod n".

Figure 11-15. Modular exponentiation of normal mode



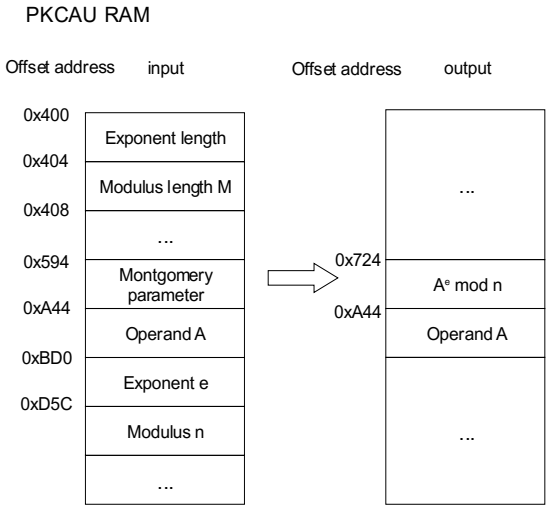
0<L<=3136, 0<M<=3136, 0<=A<n, 0<=e<2^L, 0<=result<n, 1<n<2^M (n must be odd integer).

Fast mode

The Modular exponentiation of fast mode operation is selected by configuring MODSEL[5:0]

in PKCAU_CTL register as "000010". The operation declaration is shown in ***Figure 11-16. Modular exponentiation of fast mode***. The operation result is "result = A^e mod n".

Figure 11-16. Modular exponentiation of fast mode

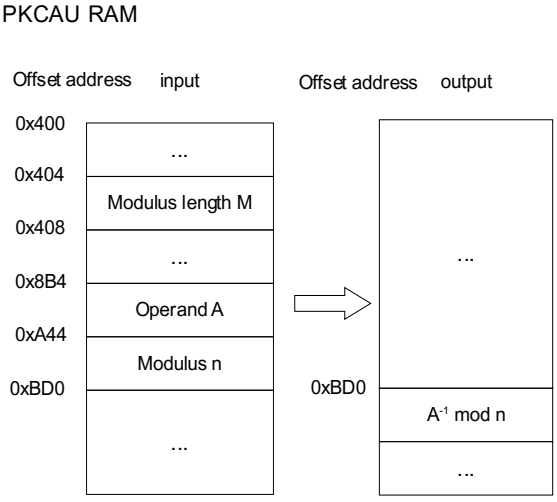


0<=A<n, 0<=e<n, 0<=result<n, 0<n<2^M, 0<M<=3136, 0<Montgomery parameter (R^2 mod n)<n.

Modular inversion

The Modular exponentiation of fast mode operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "001000". The operation declaration is shown in ***Figure 11-17. Modular inversion***. The operation result is "result = A^{-1} mod n".

Figure 11-17. Modular inversion



0<A<n, 0<result<n, 0<n<2^M, 0<M<=3136.

Note:

1. If the modulus n is prime, for all values of A that satisfy the condition "1<=A<n", the modular inversion output is valid.

2. If the modulus n is not prime, only when the greatest common divisor of A and n is 1, the modular inversion output is valid.

RSA CRT exponentiation

The RSA CRT exponentiation operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "000111".

p and q are part of the private key, and are primes

$d_P = d \bmod (p-1)$

$d_Q = d \bmod (q-1)$

$q_{inv} = q^{-1} \bmod p$

These parameters above allow the recipient to compute the exponentiation $m = A^d \pmod{pq}$ more efficiently as follows:

$m = A^d \pmod{pq}$

$m_1 = A^{dP} \bmod p$

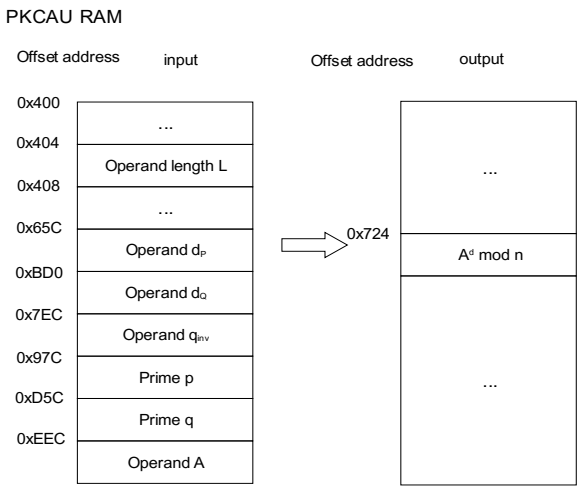
$m_2 = A^{dQ} \bmod p$

$h = q_{inv}(m_1 - m_2) \bmod p,\ m_1 > m_2$

$m = m_2 + hq$

The operation declaration is shown in ***Figure 11-18. RSA CRT exponentiation***. The operation result is "result = A^d mod pq".

Figure 11-18. RSA CRT exponentiation



The range of parameters used by RSA CRT exponentiation operation is shown in ***Table 11-3. Range of parameters used by RSA CRT exponentiation operation***.

Table 11-3. Range of parameters used by RSA CRT exponentiation operation

Parameters		Range
Input	Operand d_P	$0<=d_P<2^{L/2}$
	Operand d_Q	$0<=d_Q<2^{L/2}$
	Operand q_{inv}	$0<q_{inv}<2^{L/2}$
	Prime p	$0<p<2^{L/2}$
	Prime q	$0<q<2^{L/2}$
	Operand A	$0<=A<2^{L}$
Output	result = A^d mod pq	0<=result<pq

11.3.5. Elliptic curve operations in Fp domain

The Elliptic curve operation mode in Fp domain can be selected by configuring the MODSEL[5:0] in PKCAU_CTL register. The operation modes to be selected is shown in ***Table 11-4. Elliptic curve operations in Fp domain***.

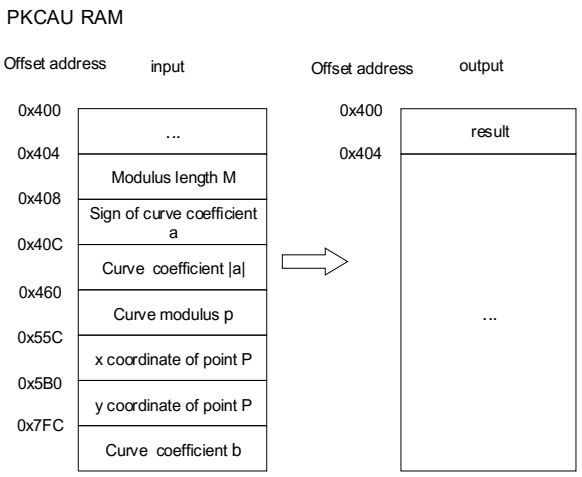
Table 11-4. Elliptic curve operations in Fp domain

MODSEL[5:0]	Operation modes
100000	Montgomery parameter computation then ECC scalar multiplication
100010	ECC scalar multiplication only (Montgomery parameter must be loaded first)
100100	ECDSA sign
100110	ECDSA verification
101000	Point on elliptic curve Fp check

Point on elliptic curve Fp check

The operation is used to check whether P(x,y) is on the $y^2= x^3+ax+b \bmod p$ in prime domain, where A and B are curve coefficients. The point on elliptic curve check operation in Fp domain is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "101000". The operation declaration is shown in ***Figure 11-19. Point on elliptic curve Fp check***. If the operation result is 0, it indicates that point P is on the elliptic curve. Or else, it indicates that point P is not on the elliptic curve.

Figure 11-19. Point on elliptic curve Fp check



The range of parameters used by point on elliptic curve Fp check operation is shown in ***Table 11-5. Range of parameters used by point on elliptic curve Fp check***.

Table 11-5. Range of parameters used by point on elliptic curve Fp check

Parameters		Range
Input	Modulus length M	0<M<=640
	Sign of curve coefficient a	0x0: positive 0x1: negative
	Curve coefficient |a|	Absolute value |a|<p
	Curve coefficient b	Absolute value |b|<p
	Curve modulus p	Odd prime 0<p<=2^M
	x coordinate of point P	x<p
	y coordinate of point P	y<p

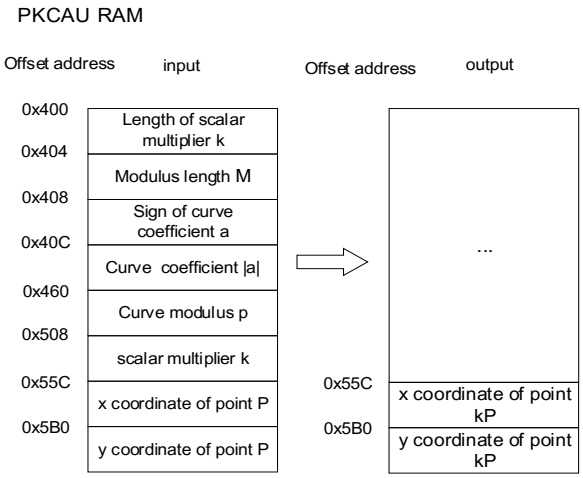
ECC scalar multiplication

ECC scalar multiplication operation is ak×P(x_P,y_P), where P is a point on the elliptic curve in the prime domain Fp. The operation result is also on the elliptic curve, or a point at infinity.

Normal mode

The ECC scalar multiplication operation in normal mode is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "100000". The operation declaration is shown in ***Figure 11-20. ECC scalar multiplication of normal mode***.

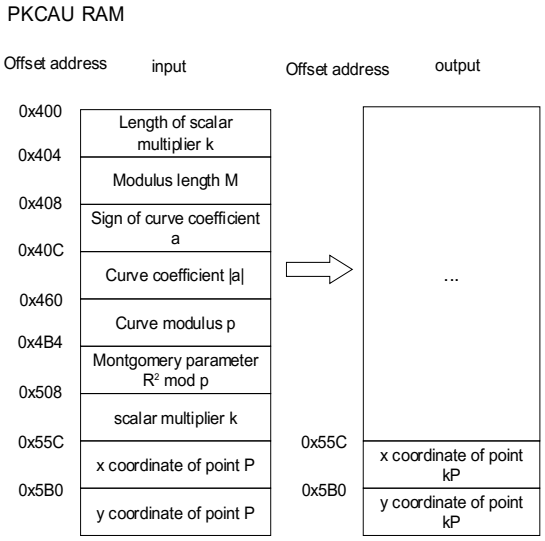
Figure 11-20. ECC scalar multiplication of normal mode



Fast mode

The ECC scalar multiplication operation in fast mode is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "100010". The operation declaration is shown in ***Figure 11-21. ECC scalar multiplication of fast mode***.

Figure 11-21. ECC scalar multiplication of fast mode



The range of parameters used by point on elliptic curve Fp check operation is shown in ***Table 11-6. Range of parameters used by ECC scalar multiplication***.

Table 11-6. Range of parameters used by ECC scalar multiplication

Parameters		Range
input	Length of scalar multiplier k (LEN)	0<LEN<=640
	Modulus length M	0<M<=640
	Sign of curve coefficient a	0x0: positive 0x1: negative
	Curve coefficient |a|	Absolute value |a|<p
	Curve modulus p	Odd prime 0<p<=2^M
	scalar multiplier k	0<=k<2^{LEN} (k < n, and n is the curve prime order)
	x coordinate of point P	x_P<p
	y coordinate of point P	y_P<p
output	x coordinate of point kP	x<p
	y coordinate of point kP	y<p

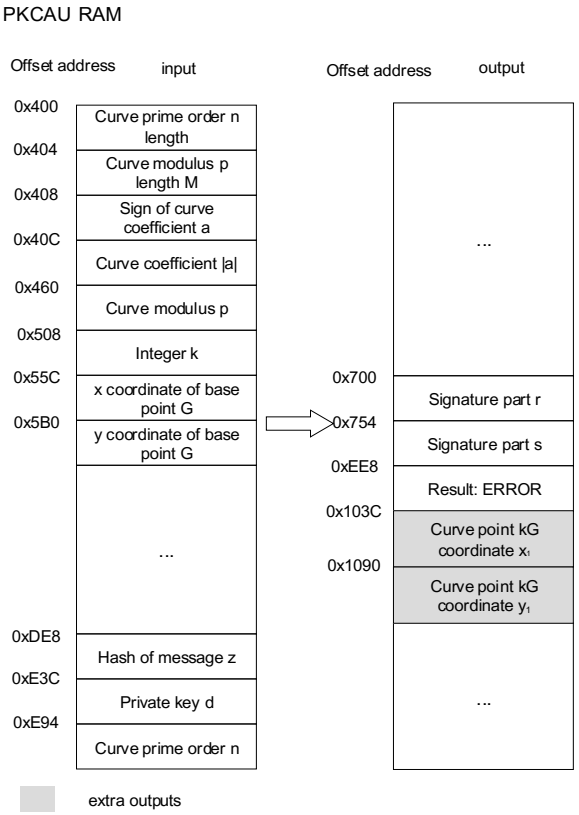
If k=0, the output is a point at infinity. When k is a multiple of curve prime order n, the output will also be a point at infinity. In this module, output is (0, 0) if the result is a point at infinity.

If k<0, the absolute value of k replaces k as the scalar multiplier for ECC scalar multiplication. After the computation is completed, -P = (x, -y) can be used to compute the final result of y.

ECDSA sign

The ECDSA sign operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "100100". The operation declaration is shown in ***Figure 11-22. ECDSA sign***.

Figure 11-22. ECDSA sign



The range of parameters used ECDSA sign operation is shown in ***Table 11-7. Range of parameters used by ECDSA sign***.

Table 11-7. Range of parameters used by ECDSA sign

Parameters		Range
input	Curve prime order n length (LEN)	0<LEN<=640
	Curve modulus p length (M)	0<M<=640
	Sign of curve coefficient a	0x0: positive 0x1: negative
	Curve coefficient |a|	Absolute value |a|<p
	Curve modulus p	Odd prime 0<p<2^M
	Integer k	0<=k<2^{LEN}
	x coordinate of base point G	x<p
	y coordinate of base point G	y<p
	Hash of message z	z<2^{LEN}
	Private key d	positive integer d < n
	Curve prime order n	prime n<2^{LEN}
output	Signature part r	0<r<n

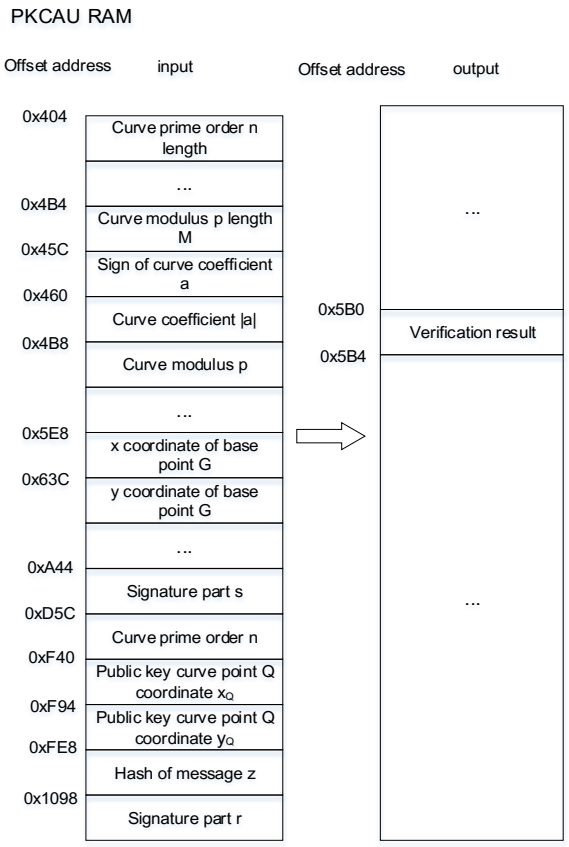
Parameters		Range
	Signature part s	0<s<n
	Signature result: ERROR	0x0: no error 0x1: Signature part r is 0 0x2: Signature part s is 0
	Curve point kG coordinate x_1	0<=x_1<n
	Curve point kG coordinate y_1	0<=y_1<n

If the error output is not 0, the content in PKCAU RAM will be cleared by hardware to avoid leaking private key related information.

ECDSA verification

The ECDSA verification operation is selected by configuring MODSEL[5:0] in PKCAU_CTL register as "100110". The operation declaration is shown in *Figure 11-23. ECDSA verification*.

Figure 11-23. ECDSA verification



The range of parameters used ECDSA verification operation is shown in *Table 11-8. Range of parameters used by ECDSA verification*.

Table 11-8. Range of parameters used by ECDSA verification

Parameters		Range
input	Curve prime order n length (LEN)	0<LEN<=640
	Curve modulus p length (M)	0<M<=640
	Sign of curve coefficient a	0x0: positive 0x1: negative
	Curve coefficient |a|	Absolute value |a|<p
	Curve modulus p	Odd prime 0<p<2^M
	x coordinate of base point G	x<p
	y coordinate of base point G	y<p
	Public key curve point Q coordinate x_Q	x_Q<p
	Public key curve point Q coordinate y_Q	y_Q<p
	Signature part r	0<r<n
	Signature part s	0<s<n
	Hash of message z	z<2^{LEN}
	Curve prime order n	prime n<2^{LEN}
output	verification result	0x0: verification valid Not 0x0: verification invalid

11.3.6. PKCAU operation process

The PKCAU can be enabled by setting the PKCAUEN bit in PKCAU_CTL register. When the PKCAU is performing a calculation, the PKCAUEN should not be cleared, or else the ongoing operation is terminated and the content in PKCAU RAM will not be guaranteed.

When PKCAUEN is 0, the application can still access PKCAU RAM through the AHB interface.

Operation process in normal mode

The flows below applies to all operations listed in MODSEL[5:0] bits in the PKCAU_CTL register.

1. After system reset, the PKCAU RAM is to be erased from the beginning to the end. During this period, the BUSY bit in PKCAU_STAT register is set. All operation to PKCAU RAM should not carry out until the BUSY bit is 0;
2. Load the initial data into PKCAU RAM at offset address 0x400;

3. Specify the operation to be performed in MODSEL[5:0] bits in PKCAU_CTL register, then set START bit in PKCAU_CTL register;
4. Wait for the ENDF bit set in PKCAU_STAT register;
5. Read calculation result from the PKCAU RAM, then clear the ENDF bit by setting the ENDFC bit in PKCAU_STATC register.

Operation process in fast mode

The fast mode computes the Montgomery parameter only once when calculating many operations with the same modulus. When the operation is performed, the pre-calculated Montgomery parameter is loaded to perform the calculation.

The flow of operation in fast mode is as follows:

1. Load the initial data into PKCAU RAM at offset address 0x400;
2. Select the Montgomery parameter calculation mode by configuring the MODSEL[5:0] as "000001", then set START bit in PKCAU_CTL register;
3. Wait for the ENDF bit set in PKCAU_STAT register;
4. Read the Montgomery parameter from the PKCAU RAM, then clear the ENDF bit by setting the ENDFC bit in PKCAU_STATC register;
5. Load the initial data and Montgomery parameter into PKCAU RAM;
6. Specify the operation to be performed in MODSEL[5:0] bits in PKCAU_CTL register, then set START bit in PKCAU_CTL register;
7. Wait for the ENDF bit set in PKCAU_STAT register;
8. Read calculation result from the PKCAU RAM, then clear the ENDF bit by setting the ENDFC bit in PKCAU_STATC register.

11.3.7. Processing times

The following tables summarize the PKCAU computation times, expressed in clock cycles.

Table 11-9. Modular exponentiation computation times

Exponent length (in bits)	Mode	Operand length (in bits)		
		1024	2048	3072
1024	Normal	6780000	-	-
	Fast	6701000	-	-
	CRT	1853000	-	-
2048	Normal	-	52196000	-
	Fast	-	51910000	-
	CRT	-	13651000	-
3072	Normal	-	-	182783000

Exponent length (in bits)	Mode	Operand length (in bits)		
		1024	2048	3072
	Fast	-	-	181953000
	CRT	-	-	44905000

Table 11-10. ECC scalar multiplication computation times

Mode	Modulus length (in bits)					
	160	192	256	320	384	512
Normal	626000	951000	1997000	3617000	5762000	13134000
Fast	623000	946000	1990000	3607000	5749000	13111000

Table 11-11. ECDSA signature average computation times

Modulus length (in bits)					
160	192	256	320	384	512
634000	966000	2029000	3648000	5833000	13177000

Table 11-12. ECDSA verification average computation times

Modulus length (in bits)					
160	192	256	320	384	512
1261000	1901000	3997000	7225000	11477000	26287000

Table 11-13. Montgomery parameters average computation times

Modulus length (in bits)								
160	192	256	320	384	512	1024	2048	3072
3873	4658	7109	10330	14526	22301	79116	284359	626909

11.3.8. Status, errors and interrupts

There are several status and error flags in PKCAU, and interrupt may be asserted from these flags by setting some register bits.

- Access address error (ADDRERR)

When the accessed address exceeds the expected range of PKCAU RAM, the address error flag ADDRERR bit in PKCAU_STAT register will be set. If the ADDRERRIE bit is set in PKCAU_CTL register, an interrupt will be generated. The ADDRERR bit can be cleared by setting the ADDRERRC bit in PKCAU_STATC register.

- RAM error (RAMERR)

The PKCAU core is using the PKCAU RAM while an AHB access to the PKCAU RAM occurs, the RAM error flag RAMERR bit in PKCAU_STAT register will be set. If it is an AHB read, it returns 0, an AHB write will be ignored. If the RAMERRIE bit is set in PKCAU_CTL register, an interrupt will be generated. The RAMERR bit can be cleared by setting the RAMERRC bit in PKCAU_STATC register.

- End of operation flag (ENDF)

When the operation specified in MODSEL[5:0] bits in the PKCAU_CTL register is completed,

the ENDF bit will be set. If the ENDIE bit in PKCAU_CTL register is set, an interrupt will be generated. The ENDF bit can be cleared by setting the ENDFC bit in PKCAU_STATC register. The ENDF bit can also be cleared automatically if another operation is carried out by set the START bit.

The PKCAU interrupt events and flags are listed in *Table 11-14. PKCAU interrupt requests*.

Table 11-14. PKCAU interrupt requests

Interrupt event	Event flag	Flag clear	Enable control bit
Access address error	ADDRERR	ADDRERRC	ADDRERRIE
RAM error	RAMERR	RAMERRC	RAMERRIE
Operation end flag	ENDF	ENDFC	ENDIE

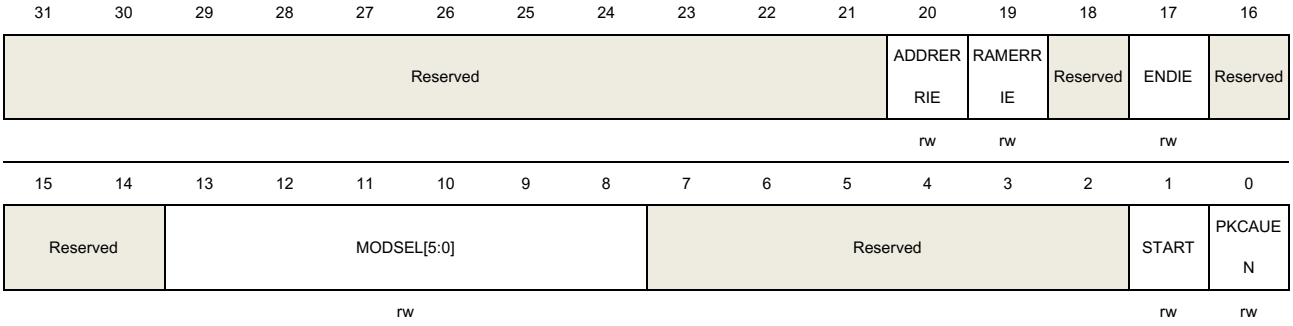
11.4. Register definition

PKCAU base address: 0x5006 1000

11.4.1. Control register (PKCAU_CTL)

Address offset: 0x00
Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20	ADDRERRIE	Address error interrupt enable 0: Address error interrupt disable 1: Address error interrupt enable
19	RAMERRIE	RAM error interrupt enable 0: RAM error interrupt disable 1: RAM error interrupt enable
18	Reserved	Must be kept at reset value.
17	ENDIE	End of operation interrupt enable 0: End of operation interrupt disable 1: End of operation interrupt enable
16:14	Reserved	Must be kept at reset value.
13:8	MODSEL[5:0]	PKCAU operation mode selection 000000: Montgomery parameter computation then modular exponentiation 000001: Montgomery parameter computation only 000010: Modular exponentiation only (Montgomery parameter must be loaded first) 000111: RSA CRT exponentiation 001000: Modular inversion 001001: Arithmetic addition 001010: Arithmetic subtraction

		001011: Arithmetic multiplication 001100: Arithmetic comparison 001101: Modular reduction 001110: Modular addition 001111: Modular subtraction 010000: Montgomery multiplication 100000: Montgomery parameter computation then ECC scalar multiplication 100010: ECC scalar multiplication only (Montgomery parameter must be loaded first) 100100: ECDSA sign 100110: ECDSA verification 101000: Point on elliptic curve Fp check Other values are reserved.
7:2	Reserved	Must be kept at reset value.
1	START	PKCAU starts operation This bit is set by software to start the PKCAU operation which is specified in MODSEL[5:0] in PKCAU_CTL register. When the BUSY bit in PKCAU_STAT register is 1, writing 1 to this bit will be ignored.
0	PKCAUEN	PKCAU enable 0: PKCAU disable 1: PKCAU enable

11.4.2. Status register (PKCAU_STAT)

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
Reserved											ADDRERR	RAMERR	Reserved	ENDF	BUSY
											r	r		r	r

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

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20	ADDRERR	Address error. 0: No address error. 1: The accessed address exceeds the expected range of PKCAU RAM, an address

		error occurs.
19	RAMERR	PKCAU RAM error 0: No PKCAU RAM error. 1: When the PKCAU core is using the RAM, AHB accesses the PKCAU RAM, a PKCAU RAM error occurs.
18	Reserved	Must be kept at reset value.
17	ENDF	End of PKCAU operation When the operation executed completely, this bit is set by hardware.
16	BUSY	Busy flag When the START bit in PKCAU_CTL register is set, this bit is set by hardware. When the PKCAU operation is completed, this bit is cleared by hardware.
15:0	Reserved	Must be kept at reset value.

11.4.3. Status clear register (PKCAU_STATC)

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
Reserved											ADDRERRC	RAMERRC	Reserved	ENDFC	Reserved
											w	w		w	

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	ADDRERRC	Address error flag clear. Software can clear the ADDRERR bit in PKCAU_STAT by writing 1 to this bit.
19	RAMERRC	PKCAU RAM error flag clear. Software can clear the RAMERR bit in PKCAU_STAT by writing 1 to this bit.
18	Reserved	Must be kept at reset value.
17	ENDFC	End of PKCAU operation flag clear. Software can clear the ENDF bit in PKCAU_STAT by writing 1 to this bit.
16:0	Reserved	Must be kept at reset value.

12. Hash Acceleration Unit (HAU)

12.1. 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 180-2 (FIPS PUB 180-2).
- Secure Hash Standard specifications (SHA-1, SHA-224, SHA-256).
- Internet Engineering Task Force Request for Comments number 1321 (IETF RFC 1321) specifications (MD5).

12.2. 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.

12.3. 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 12-1. DATAM No swapping and Half-word swapping and *Figure 12-2. DATAM Byte swapping and Bit swapping* illustrate the data swapping according to different data types.

Figure 12-1. DATAM No swapping and Half-word swapping

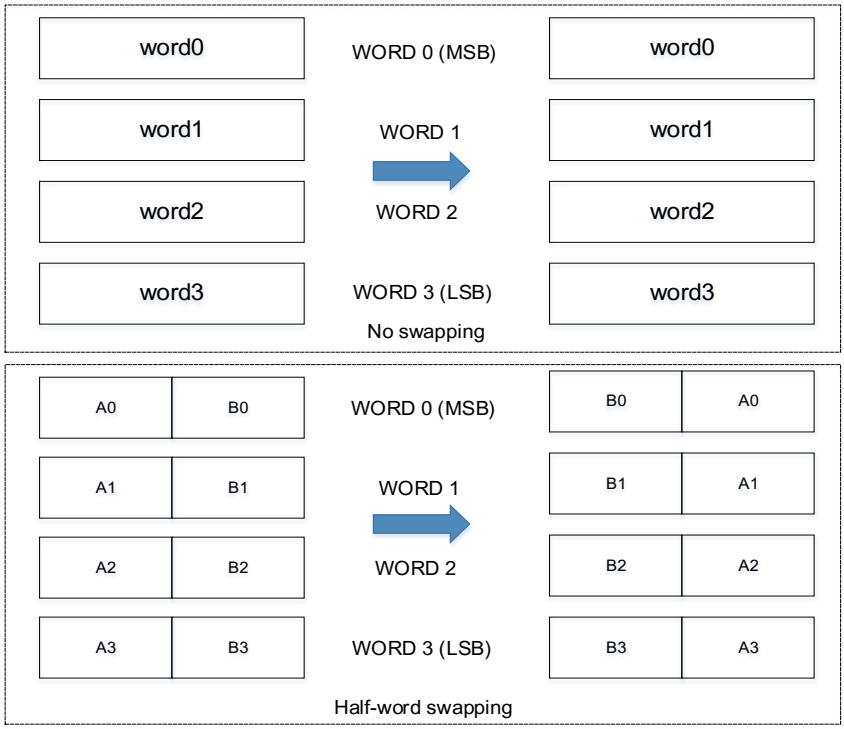
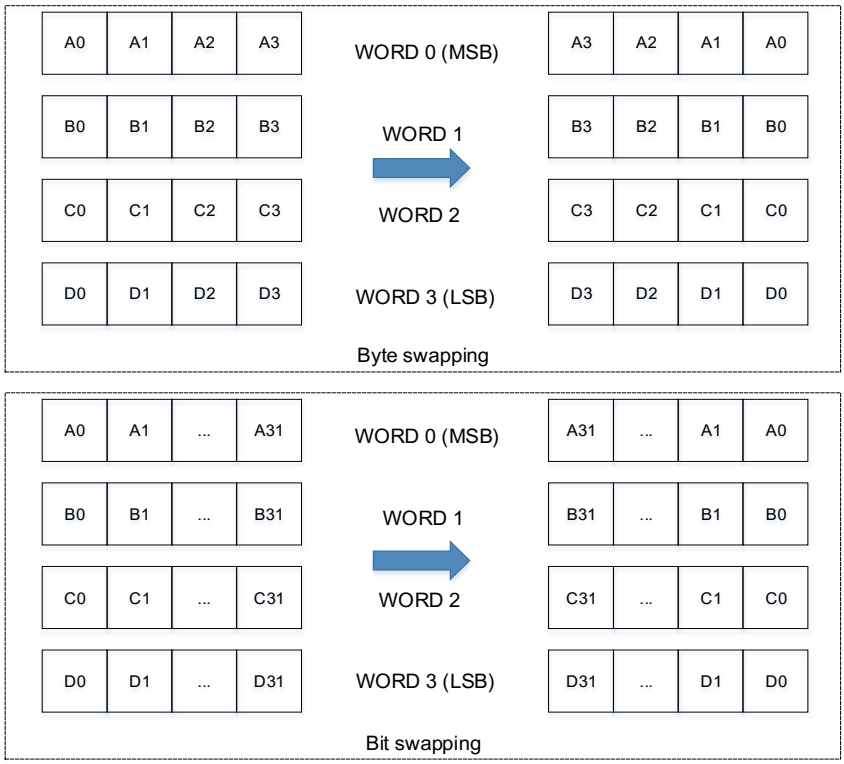


Figure 12-2. DATAM Byte swapping and Bit swapping

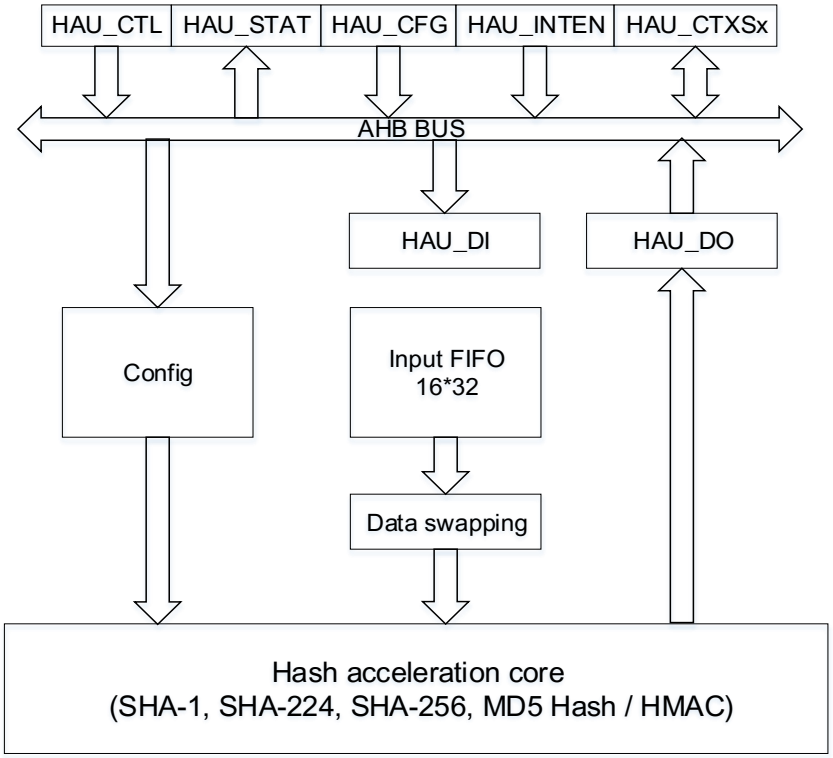


12.4. HAU core

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 12-3. HAU block diagram



12.4.1. **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

12.4.2. 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.

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.

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.

12.4.3. Hash mode

The hash mode is selected when the HMS bit in the HAU_CTL register is set as 0. And when the START bit in the HAU_CTL register is 1, SHA-1, SHA-224, SHA-256 and MD5 mode computation is chosen by the ALGM bits.

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.

12.4.4. 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 | 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 as 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 set 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.

12.5. 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.

12.5.1. 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.

12.5.2. 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 DMA1 channel 7 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.

12.6. 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.

12.6.1. **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.

12.6.2. 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.

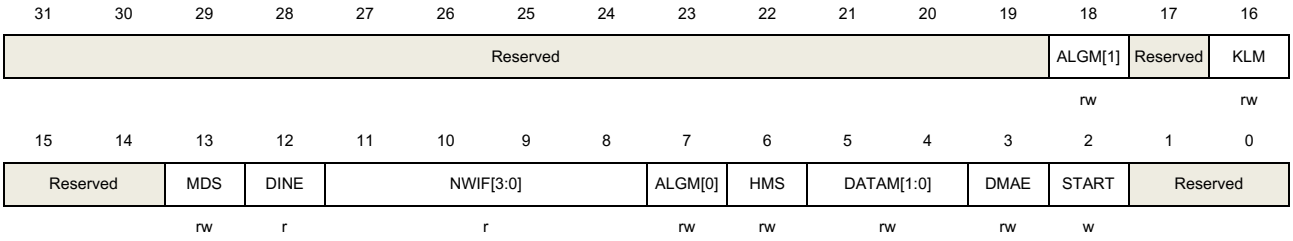
12.7. Register definition

HAU base address: 0x5006 0400

12.7.1. HAU control register (HAU_CTL)

Address offset: 0x00
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	ALGM[1]	Algorithm selection bit 1
17	Reserved	Must be kept at reset value.
16	KLM	Key length mode 0: Key length <= 64 bytes 1: Key length > 64 bytes **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 Set this bit if hash message is large files and multiple DMA transfers are needed. 0: Single DMA transfers needed and CALEN bit is automatically set at the end of a DMA transfer 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 0: The DI register is empty 1: The DI register is not empty **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 **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: 00: Select SHA-1 algorithm 01: Select MD5 algorithm 10: Select SHA224 algorithm 11: Select SHA256 algorithm
6	HMS	HAU mode selection, must be changed when no computation is processing 0: HASH mode selected 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 Defines the format of the data entered into the HAU_DI register: 00: No swapping. The data written to HAU_DI is direct write to FIFO without swapping. 01: Half-word swapping. The data written into HAU_DI need half-word swapping before write to FIFO. 10: Bytes swapping. The data written into HAU_DI need bytes swapping before write to FIFO. 11: Bit swapping. The data written into HAU_DI need bytes swapping before write to FIFO.
3	DMAE	DMA enable 0: DMA disabled 1: DMA enabled **Note:** 1. This bit is cleared when transferring the last data of the message, but not cleared because of START. 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 1: Start the digest of a new message 0: No effect **Note:** Reading this bit always returns 0.
1:0	Reserved	Must be kept at reset value.

12.7.2. HAU 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 When write to these registers, the current content pushed to IN FIFO and new value updates. When read, returns the current content.

12.7.3. HAU 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 0: No calculation 1: Start data padding with VBL prepared previously. Start the calculation of the last digest **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 0x00: All 32 bits of the last data written to HAU_DI after data swapping are valid 0x01: Only bit [31] of the last data written to HAU_DI after data swapping are valid 0x02: Only bits [31:30] of the last data written to HAU_DI after data swapping are valid 0x03: Only bits [31:29] of the last data written to HAU_DI after data swapping are valid ... 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.

12.7.4. HAU data output register (HAU_DO0..7)

Reset value: 0x0000 0000
This register has to be accessed by word (32-bit)

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

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

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

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

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

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

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

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

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

12.7.5. HAU 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 0: Calculation completion interrupt is disabled 1: Calculation completion interrupt is enabled
0	DIIE	Data input interrupt enable 0: Data input interrupt is disabled 1: Data input interrupt is enabled

12.7.6. HAU 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 0: No processing 1: Data block is in process
2	DMAS	DMA status 0: DMA is disabled (DMAE = 0) and no transfer is processing 1: DMA is enabled (DMAE = 1) or a transfer is processing
1	CCF	Digest calculation completion flag 0: Digest calculation is not completed 1: Digest calculation is completed
0	DIF	Data input flag 0: A data is written to data input register 1: A data processing is completed (only the data in input FIFO will be processed)

12.7.7. 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.

13. Cryptographic Acceleration Unit (CAU)

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. It is fully compliant implementation of 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.

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

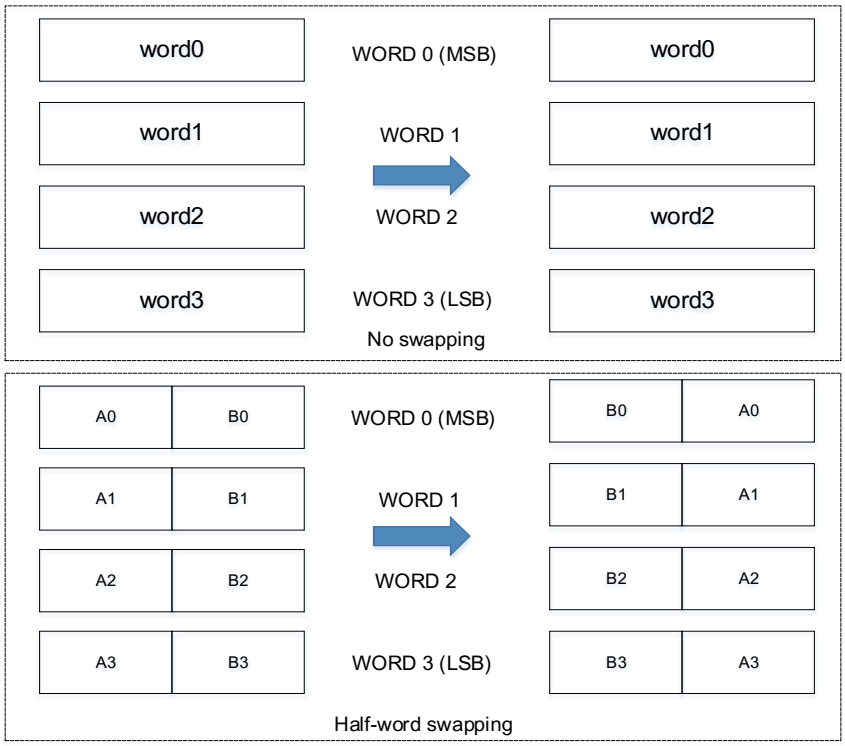
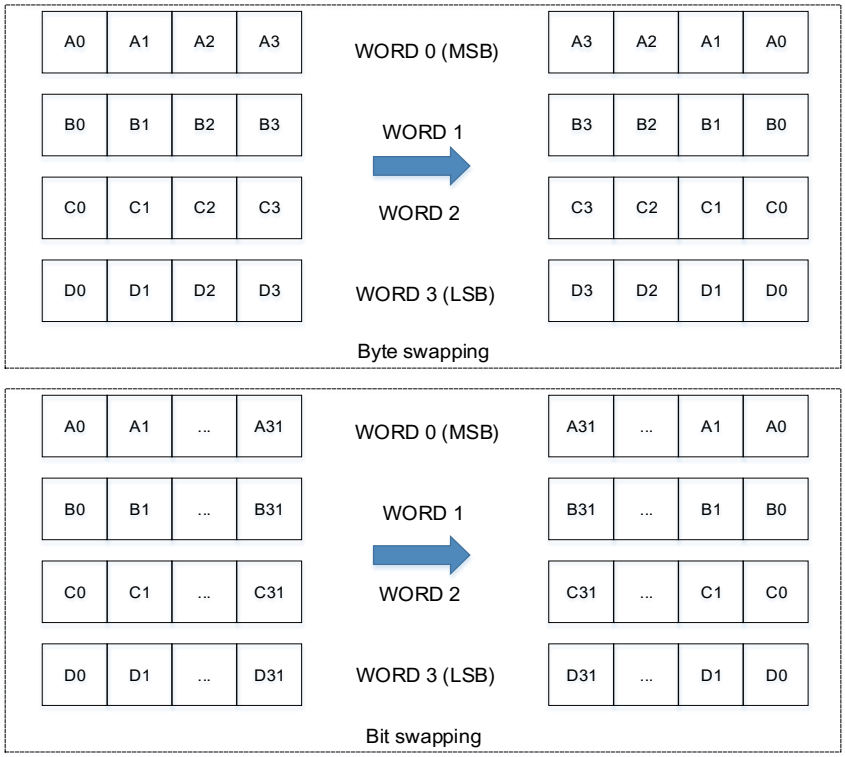


Figure 13-2. DATAM Byte swapping and Bit swapping



13.3.2. Initialization vectors

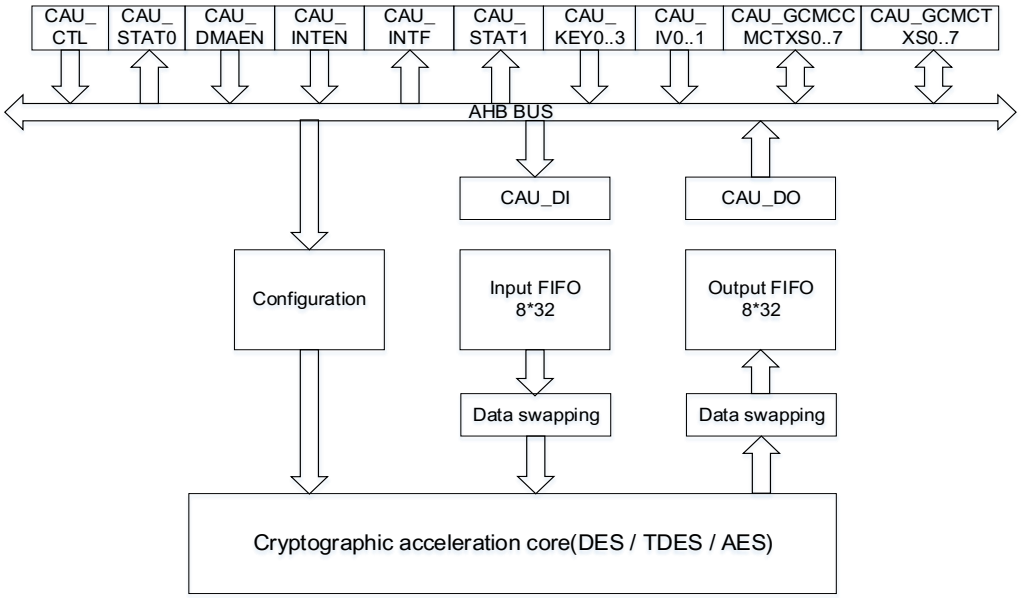
The initialization vectors are used in CBC, CTR, GCM, GMAC, CCM, CFB and OFB modes to XOR with data blocks. They are independent of plaintext and ciphertext, and the DATAM value will not affect them. Note the initialization vector registers CAU_IV0..1(H / L) can only be written when BUSY is 0, otherwise 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

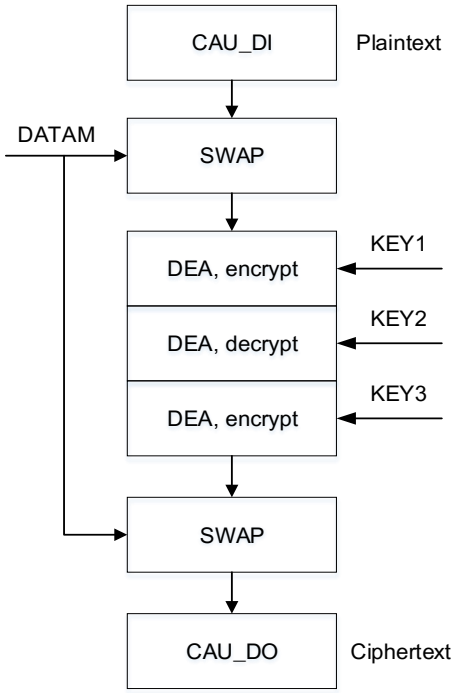
In this option, KEY1, KEY2 and KEY3 are completely independent. FIPS PUB 46-3 - 1999 (and ANSI X9.52 - 1998) refers to this option.

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***.

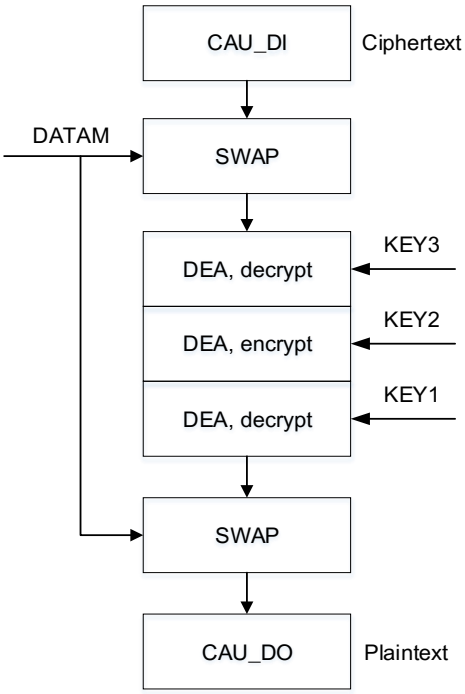
Figure 13-4. DES / TDES ECB encryption



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***.

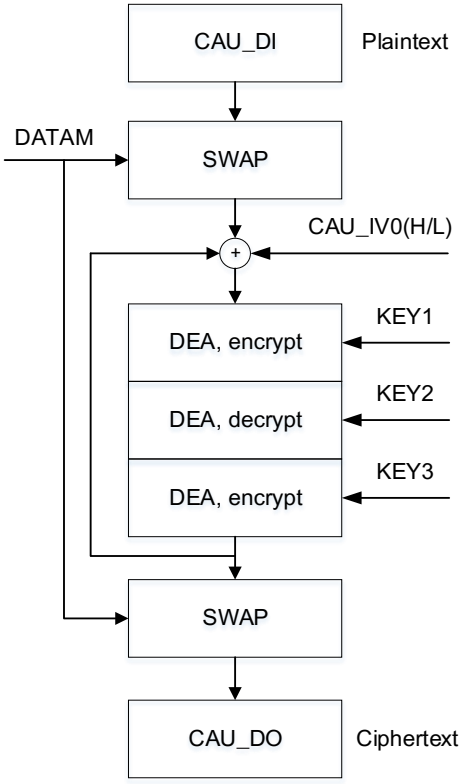
Figure 13-5. DES / TDES ECB decryption



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 if the plaintext message does not consist of an integral number of data blocks, the final partial data block should be encrypted in a specified manner. 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***.

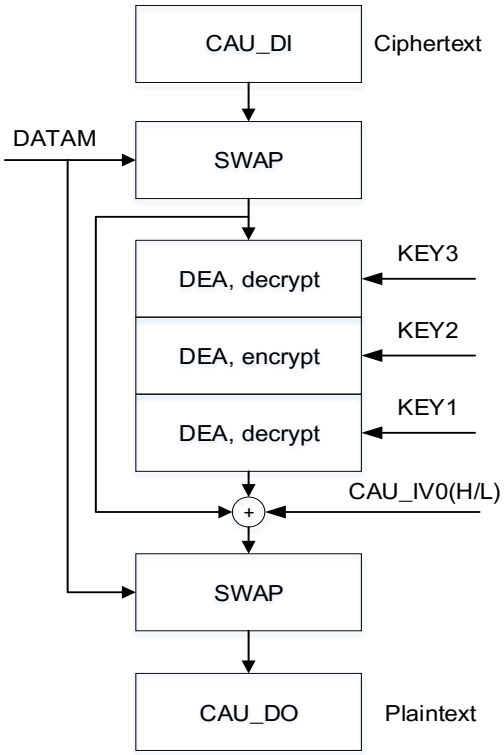
Figure 13-6. DES / TDES CBC encryption



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.

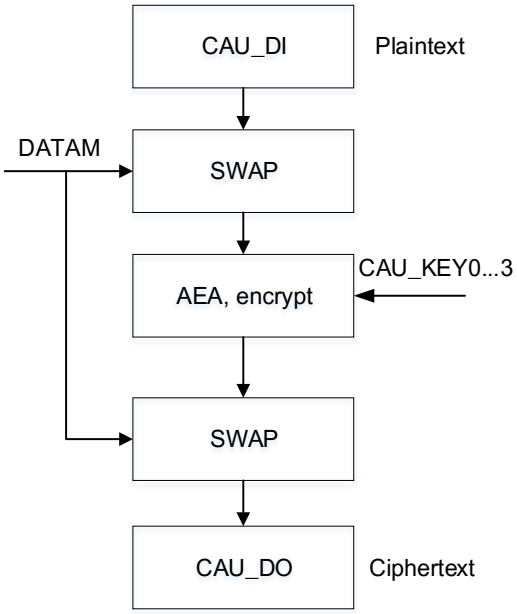
The AES key is used as [KEY3 KEY2] when the key size is configured as 128, [KEY3 KEY2 KEY1] when the key size is configured as 192 and [KEY3 KEY2 KEY1 KEY0] when the key size is configured as 256.

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***.

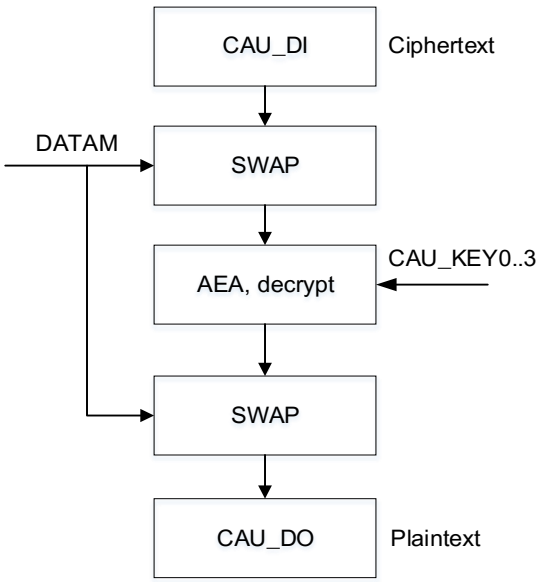
Figure 13-8. AES ECB encryption



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***.

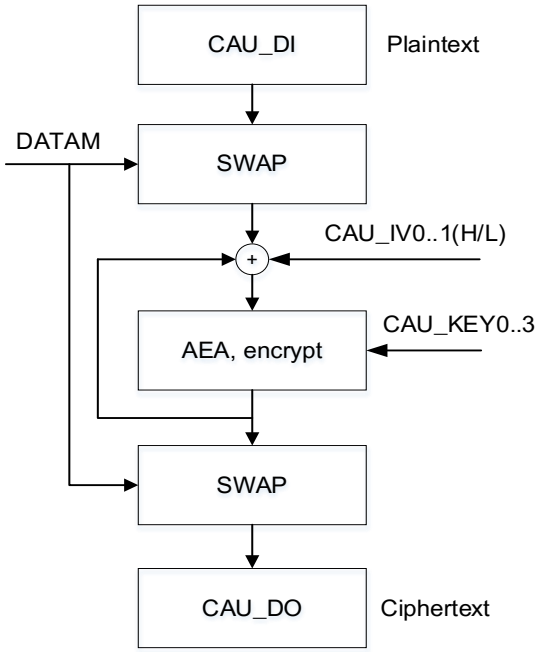
Figure 13-9. AES ECB decryption



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 if the plaintext message does not consist of an integral number of data blocks, the final partial data block should be encrypted in a specified manner. 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

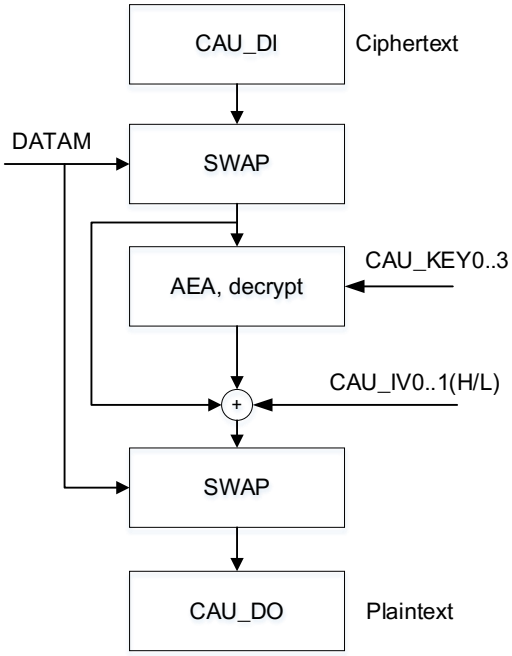


AES-CBC mode decryption

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

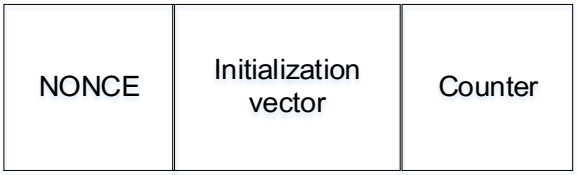
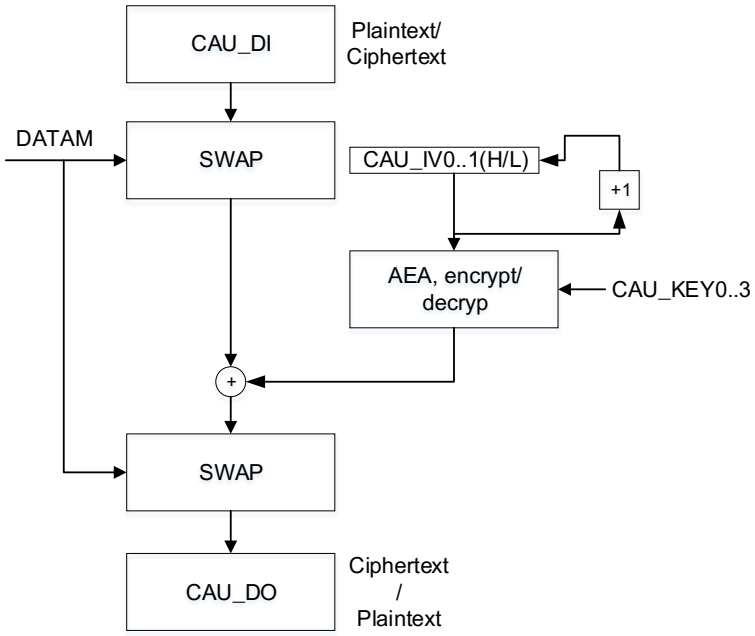


Figure 13-13. AES CTR encryption / decryption



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 in this phase.

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 header 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 vice versa.

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 vice versa.

13.5. Operating modes

Encryption

1. Disable the CAU by resetting the CAUEN bit in the CAU_CTL register.
2. Enable CAU power domain by setting the CORE1WAKE bit in the PMU_CTL1 register, and then enable CAU clock.
3. Select and configure the key length with the KEYM bits in the CAU_CTL register if AES algorithm is chosen.
4. Configure the CAU_KEY0..3(H / L) registers according to the algorithm.
5. Configure the DATAM bit in the CAU_CTL register to select the data swapping type.
6. 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.
7. Configure the encryption direction by writing 0 to the CAUDIR bit in the CAU_CTL register.
8. Configure the initialization vectors by writing the CAU_IV0..1 registers.
9. Flush the input FIFO and output FIFO by configure the FFLUSH bit in the CAU_CTL register when CAUEN is 0.
10. Enable the CAU by set the CAUEN bit as 1 in the CAU_CTL register.
11. 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.
12. Wait for ONE bit in the CAU_STAT0 register is 1 then read the CAU_DO registers. The output data can also be transferred by DMA / CPU during interrupts / no DMA or interrupts.
13. Repeat steps 10, 11 until all data blocks has been encrypted.

Decryption

1. Disable the CAU by resetting the CAUEN bit in the CAU_CTL register.
2. Enable CAU power domain by setting the CORE1WAKE bit in the PMU_CTL1 register, and then enable CAU clock.

3. Select and configure the key length with the KEYM bits in the CAU_CTL register if AES algorithm is chosen.
4. Configure the CAU_KEY0..3(H / L) registers according to the algorithm.
5. Configure the DATAM bit in the CAU_CTL register to select the data swapping type.
6. Configure the ALGM[3:0] bits to "0111" in the CAU_CTL register to complete the key derivation.
7. Enable the CAU by set the CAUEN bit as 1.
8. Wait until the BUSY and CAUEN bit return to 0 to make sure that the decryption keys are prepared.
9. 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.
10. Configure the decryption direction by writing 1 to the CAUDIR bit in the CAU_CTL register.
11. Configure the initialization vectors by writing the CAU_IV0..1 registers.
12. Flush the input FIFO and output FIFO by configure the FFLUSH bit in the CAU_CTL register when CAUEN is 0.
13. Enable the CAU by set the CAUEN bit as 1 in the CAU_CTL register.
14. 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.
15. Wait for ONE bit in the CAU_STAT0 register is 1, then read the CAU_DO registers. The output data can also be transferred by DMA / CPU during interrupts/no DMA or interrupts.
16. 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 it. It should be noted that the NBPILB[3:0] bitfield should be configured after the encryption 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.

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: 0x5006 0000

13.9.1. Control register (CAU_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								NBPILB[3:0]				ALGM[3]	Reserved	GCM_CCMPH[1:0]	
								rw				rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAUEN	FFLUSH	Reserved				KEYM[1:0]		DATAM[1:0]		ALGM[2:0]			CAUDIR	Reserved	
rw	w					rw		rw		rw			rw		

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 0000: all bytes are valid (no padding) 0001: one padding byte of last block ... 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 00: prepare phase 01: AAD phase 10: encryption/decryption phase 11: tag phase
15	CAUEN	CAU Enable 0: CAU is disabled 1: CAU is enabled **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 0: No effect 1: When CAUEN = 1, flush the input and output FIFO Reading this bit always returns 0

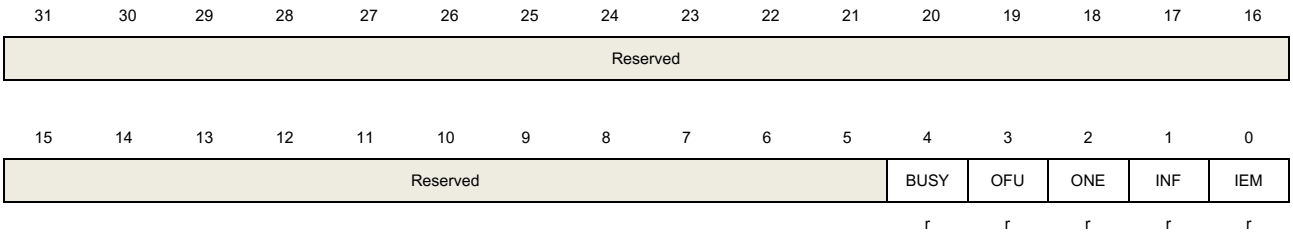
13:10	Reserved	Must be kept at reset value.
9:8	KEYM[1:0]	AES key size mode configuration, must be configured when BUSY=0 00: 128-bit key length 01: 192-bit key length 10: 256-bit key length 11: never use
7:6	DATAM[1:0]	Data swapping type mode configuration, must be configured when BUSY=0 00: No swapping 01: Half-word swapping 10: Byte swapping 11: Bit swapping
5:3	ALGM[2:0]	Encryption/decryption algorithm mode bit 0 to bit 2 These bits and bit 19 of CAU_CTL must be configured when BUSY=0 0000: TDES-ECB with CAU_KEY1, 2, 3. Initialization vectors (CAU_IV0..1) are not used 0001: TDES-CBC with CAU_KEY1, 2, 3. Initialization vectors (CAU_IV0) is used to XOR with data blocks 0010: DES-ECB with only CAU_KEY1 Initialization vectors (CAU_IV0..1) are not used 0011: DES-CBC with only CAU_KEY1 Initialization vectors (CAU_IV0) is used to XOR with data blocks 0100: AES-ECB with CAU_KEY0, 1, 2, 3. Initialization vectors (CAU_IV0..1) are not used 0101: AES-CBC with CAU_KEY0, 1, 2, 3. Initialization vectors (CAU_IV0..1) are used to XOR with data blocks 0110: AES_CTR with CAU_KEY0, 1, 2, 3. Initialization vectors (CAU_IV0..1) are used to XOR with data blocks In this mode, encryption and decryption are same, then the CAUDIR is disregarded. 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. 1000: Galois Counter Mode (GCM). This algorithm mode is also used for GMAC algorithm. 1001: Counter with CBC-MAC (CCM). 1010: Cipher Feedback (CFB) mode 1011: Output Feedback (OFB) mode
2	CAUDIR	CAU direction, must be configured when BUSY=0 0: encryption 1: decryption
1:0	Reserved	Must be kept at reset value.

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).



Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	BUSY	Busy bit 0: No processing. This is because: - CAU is disabled by CAUEN = 0 or the processing has been completed. - No enough data or no enough space in the input / output FIFO to perform a data block 1: CAU is processing data or key derivation.
3	OFU	Output FIFO is full 0: Output FIFO is not full 1: Output FIFO is full
2	ONE	Output FIFO is not empty 0: Output FIFO is empty 1: Output FIFO is not empty
1	INF	Input FIFO is not full 0: Input FIFO is full 1: Input FIFO is not full
0	IEM	Input FIFO is empty 0: Input FIFO is not empty 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 Write these bits will write data to IN FIFO, read these bits will return IN 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).

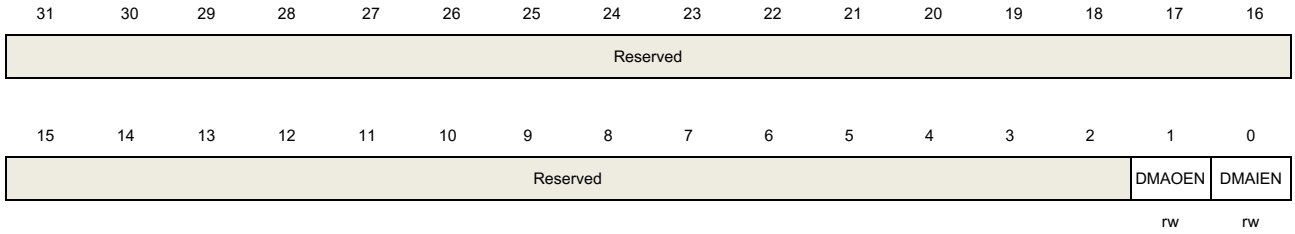
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DO[31:16]															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DO[15:0]															
r															

Bits	Fields	Descriptions
31:0	DO[31:0]	Data output These bits are read only, read these bits return OUT 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).



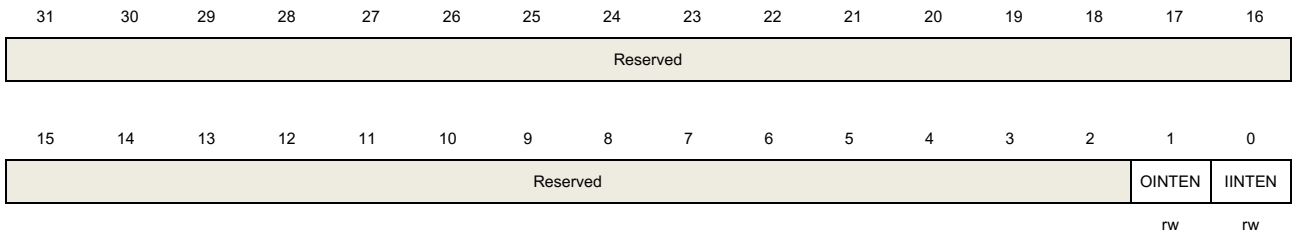
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	DMAOEN	DMA output enable 0: DMA for OUT FIFO data is disabled 1: DMA for OUT FIFO data is enabled
0	DMAIEN	DMA input enable 0: DMA for IN FIFO data is disabled 1: DMA for IN FIFO data is enabled

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).



Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	OINTEN	OUT FIFO interrupt enable 0: OUT FIFO interrupt is disable 1: OUT FIFO interrupt is enable
0	IINTEN	IN FIFO interrupt enable 0: IN FIFO interrupt is disable 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 0: OUT FIFO interrupt status not pending 1: OUT FIFO interrupt status pending
0	ISTA	IN FIFO interrupt status 0: IN FIFO interrupt not pending 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 0: OUT FIFO Interrupt not pending 1: OUT FIFO Interrupt pending
0	IINTF	IN FIFO enabled interrupt flag 0: IN FIFO Interrupt not pending 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

This 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

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

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

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

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

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

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

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

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

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

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

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

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. **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. Direct memory access controller (DMA)

14.1. 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®-M33 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.

14.2. Characteristics

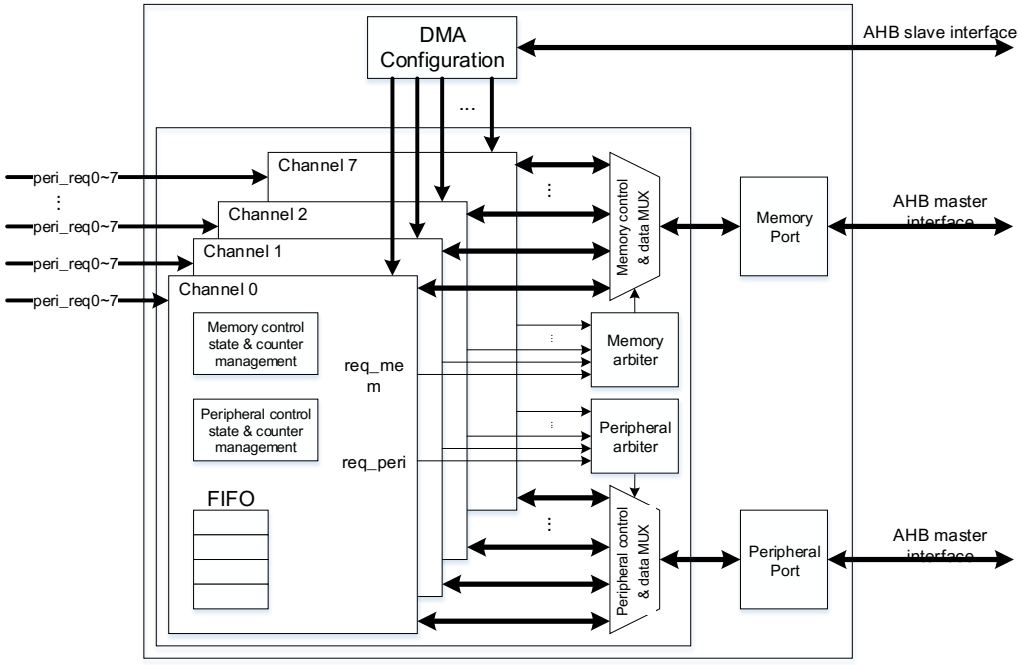
- Two AHB master interface for transferring data, and one AHB slave interface for programming DMA.
- 16 channels (8 for DMA0 and 8 for DMA1), up to 8 peripherals per channel with fixed hardware peripheral requests.
- Support independent single, 4, 8, 16-beat incrementing burst memory and peripheral transfer.
- 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 (only for DMA1).
- Both DMA and peripheral can be configured as flow controller

- DMA: Programmable length of data to be transferred, max to 65535.
- Peripheral: The last request signal given to DMA from peripheral determines the end of transfer.

- 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 write 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.

14.3. Block diagram

Figure 14-1. Block diagram of DMA



As shown in ***Figure 14-1. 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.

14.4. Function overview

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 14-1. Transfer mode***.

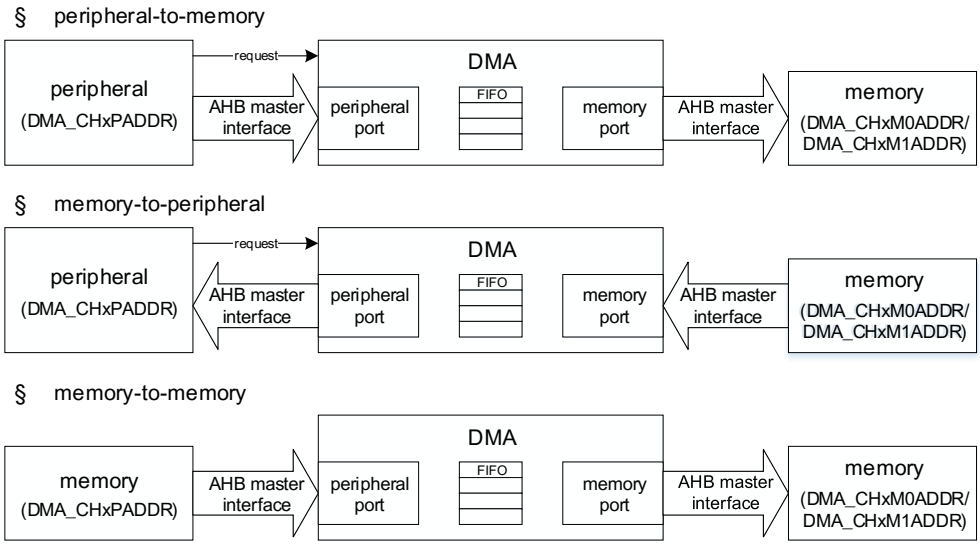
Table 14-1. Transfer mode

Transfer mode	TM[1:0]	Source	Destination
Peripheral to memory	00	DMA_CHxPADDR	DMA_CHxM0ADDR/ DMA_CHxM1ADDR1
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 14-2. Data stream for three transfer modes



As shown in ***Figure 14-2. 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.

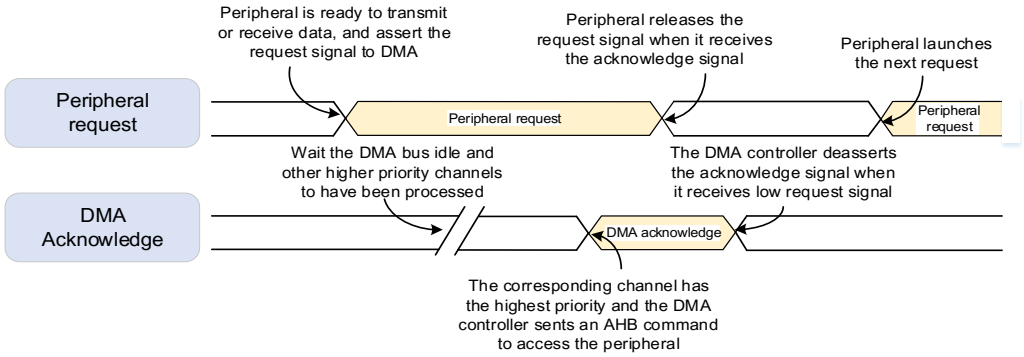
14.4.1. Peripheral handshake

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 14-3. Handshake mechanism shows how the handshake mechanism works between the DMA controller and peripherals.

Figure 14-3. Handshake mechanism



Each DMA has 8 channels, with fixed multiple peripheral requests. The PERIEN bits in the DMA_CHxCTL register determine which peripheral request signal connectes to each DMA channel. The peripheral requests mapping of DMA0 is listed in *Table 14-2. Peripheral requests to DMA0*, and the peripheral requests mapping of DMA1 is listed in *Table 14-3. Peripheral requests to DMA1*.

As listed in the *Table 14-2. Peripheral requests to DMA0* and *Table 14-3. Peripheral requests to DMA1*, a peripheral request can be connected to two different DMA channels. It is forbidden to simultaneously enable these two DMA channels with selecting the same peripheral request. For example, in DMA0, SPI2_RX is connected to channel 0 and channel 2. When the PERIEN bits in the DMA_CH0CTL register are configured to 0b000 and the PERIEN bits in the DMA_CH2CTL register are configured to 0b000, enable these two channels and responsing the request from SPI2 at the same time will cause transmission error.

Table 14-2. Peripheral requests to DMA0

Channel		Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
PERIEN[2:0]	000	SPI2_RX	I2C3_RX	SPI2_RX	SPI1_RX	SPI1_TX	SPI2_TX	I2C3_TX	SPI2_TX
	001	I2C0_RX	I2C4_RX	TIMER6_UP	I2C4_TX	TIMER6_UP	I2C0_RX	I2C0_TX	I2C0_TX
	010	TIMER3_CH0	●	I2S2_ADD_RX	TIMER3_CH1	I2S1_ADD_RX	I2S2_ADD_TX	TIMER3_UP	TIMER3_CH2

	011	I2S2_ADD_RX	TIMER1_UP TIMER1_CH2	I2C2_RX	I2S1_ADD_RX	I2C2_TX	TIMER1_CH0	TIMER1_CH1 TIMER1_CH3	TIMER1_UP TIMER1_CH3
	100	UART4_RX	USART2_RX	UART3_RX	USART2_TX	UART3_TX	USART1_RX	USART1_TX	UART4_TX
	101	UART7_TX	UART6_TX	TIMER2_CH3 TIMER2_UP	UART6_RX	TIMER2_CH0 TIMER2_TG	TIMER2_CH1	UART7_RX	TIMER2_CH2
	110	TIMER4_CH2 TIMER4_UP	TIMER4_CH3 TIMER4_TG	TIMER4_CH0	TIMER4_CH3 TIMER4_TG	TIMER4_CH1	I2C5_RX	TIMER4_UP	I2C5_TX
	111	●	TIMER5_UP	I2C1_RX	I2C1_RX	USART2_TX	DAC0_OUT0	DAC0_OUT1	I2C1_TX

Table 14-3. Peripheral requests to DMA1

Channel		Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
PERIEN[2:0]	000	ADC0	SAI0_B0	TIMER7_CH0 TIMER7_CH1 TIMER7_CH2	SAI0_B0	ADC0	SAI0_B1	TIMER0_CH0 TIMER0_CH1 TIMER0_CH2	●
	001	●	DCI	ADC1	ADC1	SAI0_B1	SPI5_TX	SPI5_RX	DCI
	010	ADC2	ADC2	●	SPI4_RX	SPI4_TX	CAU_OUT	CAU_IN	HAU_IN
	011	SPI0_RX	●	SPI0_RX	SPI0_TX	●	SPI0_TX	●	●
	100	SPI3_RX	SPI3_TX	USART0_RX	SDIO	●	USART0_RX	SDIO	USART0_TX
	101	●	USART5_RX	USART5_RX	SPI3_RX	SPI3_TX	●	USART5_TX	USART5_TX
	110	TIMER0_TG	TIMER0_CH0	TIMER0_CH1	TIMER0_CH0	TIMER0_CH3 TIMER0_TG TIMER0_CMT	TIMER0_UP	TIMER0_CH2	●
	111	●	TIMER7_UP	TIMER7_CH0	TIMER7_CH1	TIMER7_CH2	SPI4_RX	SPI4_TX	TIMER7_CH3 TIMER7_TG TIMER7_CMT

14.4.2. 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.
- Hardware priority: For channels with equal software priority level, priority is given to the channel with lower channel number.

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 control how many data to be transmitted on the channel and must be configured before enable the CHEN bit in the register. If the peripheral is configured as the flow controller, the CNT bits are forced to '0xFFFF' immediately after enabling the channel whatever the CNT bits are. 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:

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 14-4. 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 not enough for a burst transfer are transferred can be divided into single transaction automatically.

Table 14-4. 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:

a) CNT / PBURST_beats must be an integer.

b) (CNT*PWIDTH_bytes) / (MURST_beats*MWIDTH_bytes) mus be an integer.

Note:

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, 8 for INCR8 and 16 for INCR16.

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×2)/(1×16) 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×1)/(2×4) 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 32-bit 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 14-5. FIFO counter critical value configuration rules*.

Table 14-5. 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 '10'. 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 make 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.

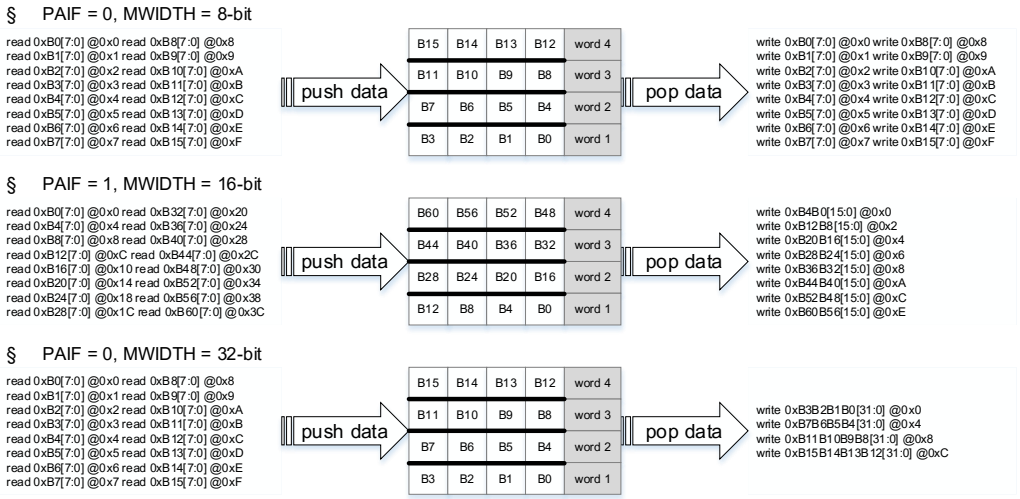
Pack/Unpack

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.

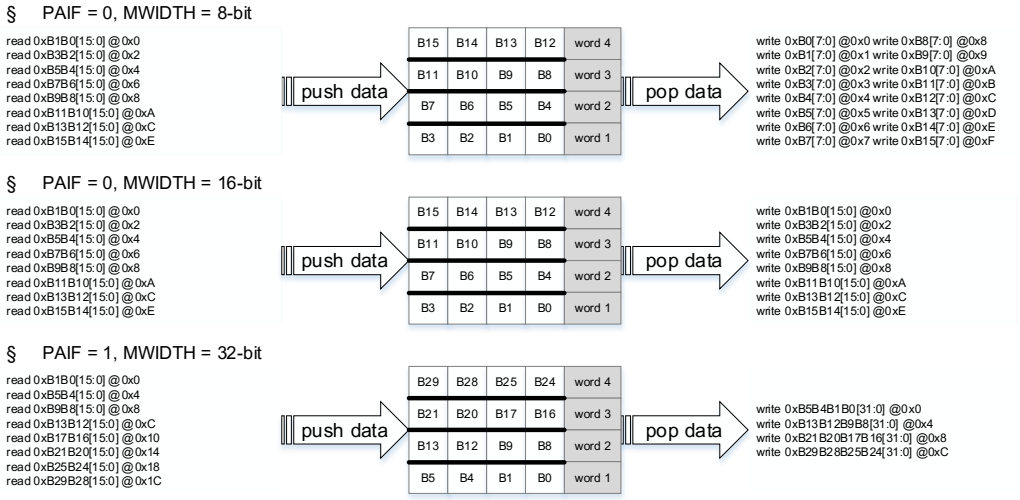
Suppose the CNT bits are 16, the PWIDTH bits are equal to '00', and both PNAGA and MNAGA are set. The DMA transfer operations for different MWIDTH are shown in the ***Figure 14-4. Data packing/unpacking when PWIDTH = '00'***.

Figure 14-4. Data packing/unpacking when PWIDTH = '00'



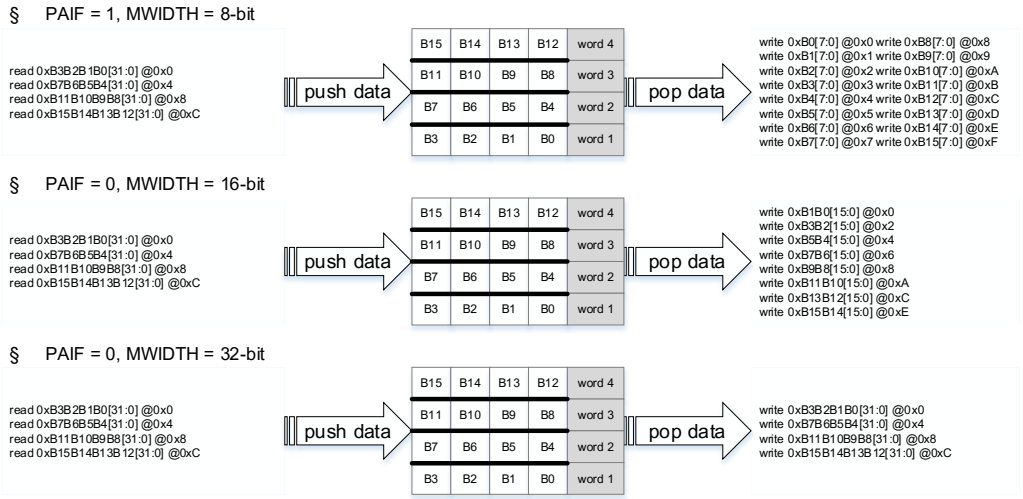
- Suppose the CNT bits are 8, the PWIDTH bits are equal to '01', and both PNAGA and MNAGA are set. The DMA transfer operations for different MWIDTH are shown in the ***Figure 14-5. Data packing/unpacking when PWIDTH = '01'***.

Figure 14-5. 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 14-6. Data packing/unpacking when PWIDTH = '10'***.

Figure 14-6. Data packing/unpacking when PWIDTH = '10'



14.4.3. 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.

14.4.4. Circular mode

Circular mode is implemented to handle continue peripheral requests. The CMEN bit in the DMA_CHxCTL register is used to enable/disable the circular mode. Circular mode is available only when DMA controls the transfer flow. When the peripheral is selected as the transfer flow controller by setting the TFCS, the circular mode is automatically disabled immediately after the channel is enabled.

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.

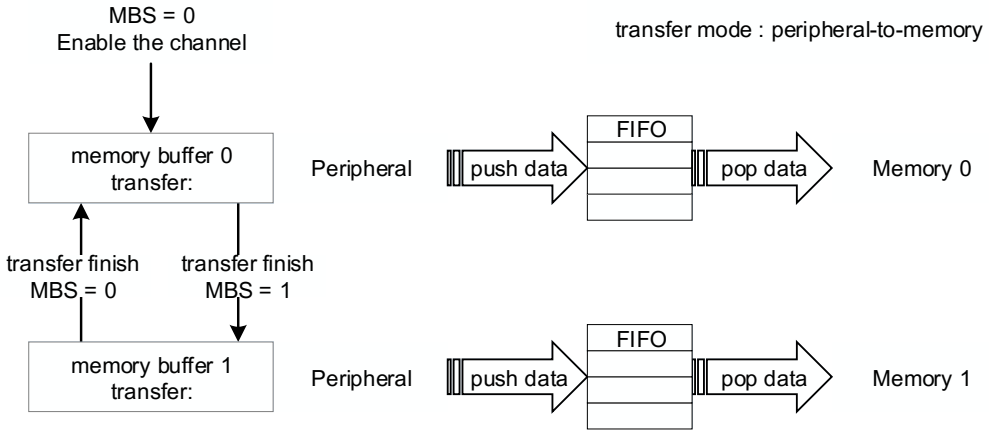
14.4.5. Switch-buffer mode

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 14-7. DMA operation of switch-buffer mode*.

Figure 14-7. DMA operation of switch-buffer mode



14.4.6. Transfer flow controller

The transfer flow controller controls the number of data items to be transferred. The TFCS bit in the DMA_CHxCTL register determines which of DMA and peripheral is selected to control the transfer flow.

- DMA as transfer flow controller: The CNT bits in the DMA_CHxCNT register determine

the number of data items to be transferred. The CNT bits must be configured before the channel is enabled.

- Peripheral as transfer flow controller: The CNT bits configured in the DMA_CHxCNT register before the channel is enabled have no meaning and these bits are force to '0xFFFF' immediately after the channel is enabled. The peripheral determines when to finish the DMA transfer by informing a last request signal to DMA.

Note: When the transfer mode is memory-to-memory, the transfer flow controller is fixed to be DMA whatever the TFCS bit is configured to.

14.4.7. 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: Only the multi-data mode is supported. 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.
- Memory-to-memory mode: Only the multi-data mode is supported. When the channel is enabled, DMA starts several peripheral transfers to fill up the FIFO. During the

transmission, the peripheral transfer is initiated once when there is enough space for it in the FIFO.

14.4.8. 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: If DMA is the transfer flow controller, 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. If peripheral is the transfer flow controller, the DMA transfer is completed when the last peripheral request has been responded and the contents of the FIFO have been entirely transferred into the memory.
- Memory-to-peripheral mode: If DMA is the transfer flow controller, when the CNT bits in the DMA_CHxCNT register reach to zero, an end of transfer is achieved. If peripheral is the transfer flow controller, the DMA transfer is completed when the last peripheral request has been responded.
- Memory-to-memory: only DMA can be the transfer flow controller. 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 and the FTFIFx bit 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***.

14.4.9. 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 PERIEN bits in the DMA_CHxCTL register to select the target peripheral. If the transfer mode is memory-to-memory, the PERIEN bits have no meaning and this step can be skipped.
5. 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, the transfer flow controller in the DMA_CHxCTL register.

6. 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.
7. 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.
8. Configure the DMA_CHxPADDR register for setting the peripheral base address.
9. If the switch-buffer mode is enabled, configure the DMA_CHxM0ADDR and DMA_CHxM1ADDR register for setting the memory base address. 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.
10. Configure the DMA_CHxCNT register to set the total transfer data number.
11. 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', 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.

14.5. 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 14-6. DMA interrupt*

events.

Table 14-6. 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
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.

14.5.1. 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 DMA is the transfer flow controller.
- When peripheral is the transfer flow controller, the last request is responded completely and the contents of the FIFO are entirely written into the memory in peripheral-to-memory mode.
- 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.

14.5.2. 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.

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.

14.5.3. 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 good 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 14-5. 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

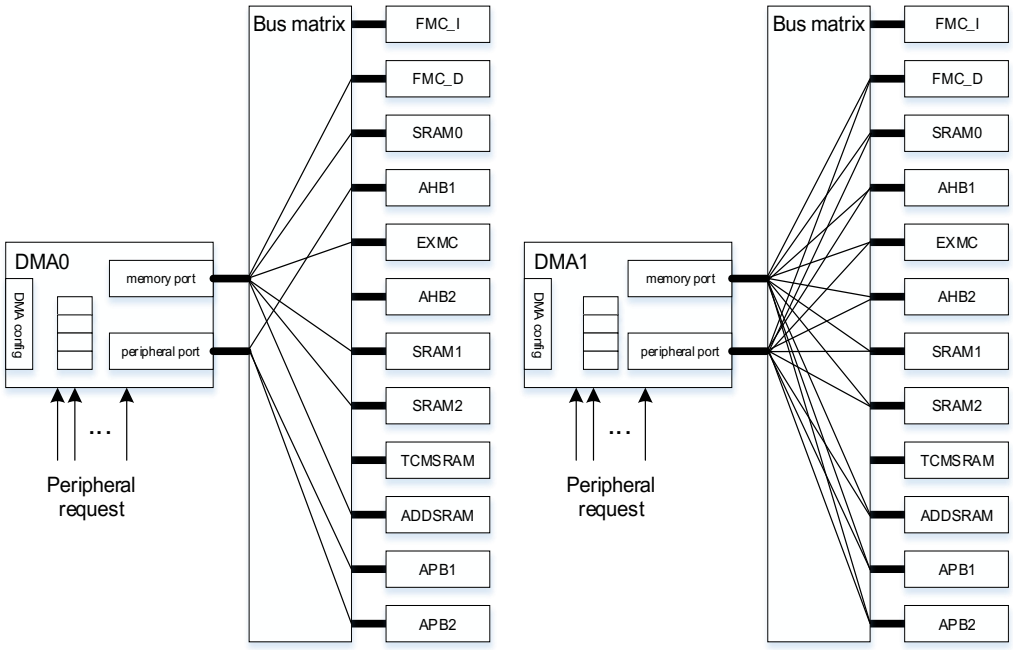
The register access error is detected only when the switch-buffer is enabled. If the software attempts to update a memory address register currently accessed by the DMA controller, a register access error is detected. For example, when the memory 0 buffer is the current source or destination, a write 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 14-8. 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 14-8. System connection of DMA0 and DMA1



14.6. Register definition

DMA0 base address: 0x4002 6000

DMA1 base address: 0x4002 6400

14.6.1. 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) Hardware set and software cleared by configuring DMA_INTC0 register. 0: Transfer has not finished on channel x 1: Transfer has finished on channel x
26/20/10/4	HTFIFx	Half transfer finish flag of channel x (x=0…3) Hardware set and software cleared by configuring DMA_INTC0 register. 0: Half number of transfer has not finished on channel x 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) Hardware set and software cleared by configuring DMA_INTC0 register. 0: Transfer access error has not occurred on channel x 1: Transfer access error has occurred on channel x
24/18/8/2	SDEIFx	Single data mode exception of channel x (x=0…3) Hardware set and software cleared by configuring DMA_INTC0 register. 0: Single data mode exception has not occurred on channel x 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) Hardware set and software cleared by configuring DMA_INTC0 register. 0: FIFO error or exception has not occurred on channel x

1: FIFO error or exception has occurred on channel x

14.6.2. 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) Hardware set and software cleared by configuring DMA_INTC1 register. 0: Transfer has not finished on channel x 1: Transfer has finished on channel x
26/20/10/4	HTFIFx	Half transfer finish flag of channel x (x=4…7) Hardware set and software cleared by configuring DMA_INTC1 register. 0: Half number of transfer has not finished on channel x 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) Hardware set and software cleared by configuring DMA_INTC1 register. 0: Transfer access error has not occurred on channel x 1: Transfer access error has occurred on channel x
24/18/8/2	SDEIFx	Single data mode exception of channel x (x=4…7) Hardware set and software cleared by configuring DMA_INTC1 register. 0: Single data mode exception has not occurred on channel x 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) Hardware set and software cleared by configuring DMA_INTC1 register. 0: FIFO error or exception has not occurred on channel x 1: FIFO error or exception has occurred on channel x

14.6.3. 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) 0: No effect 1: Clear full transfer finish flag
26/20/10/4	HTFIFCx	Clear bit for half transfer finish flag of channel x (x=0…3) 0: No effect 1: Clear half transfer finish flag
25/19/9/3	TAEIFCx	Clear bit for ransfer access error flag of channel x (x=0…3) 0: No effect 1: Clear transfer access error flag
24/18/8/2	SDEIFCx	Clear bit for single data mode exception of channel x (x=0…3) 0: No effect 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) 0: No effect 1: Clear FIFO error and exception flag

14.6.4. 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) 0: No effect 1: Clear full transfer finish flag
26/20/10/4	HTFIFCx	Clear bit for half transfer finish flag of channel x (x=4…7) 0: No effect 1: Clear half transfer finish flag
25/19/9/3	TAEIFCx	Clear bit for transfer access error flag of channel x (x=4…7) 0: No effect 1: Clear transfer access error flag
24/18/8/2	SDEIFCx	Clear bit for single data mode exception of channel x (x=4…7) 0: No effect 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) 0: No effect 1: Clear FIFO error and exception flag

14.6.5. 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				PERIEN[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	MBS	SBMEN	PRIO[1:0]	
					rw			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]		TFCS	FTFIE	HTFIE	TAEIE	SDEIE	CHEN
rw	rw		rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:25	PERIEN[2:0]	Peripheral enable Software set and cleare.

		000: Enable peripheral 0 001: Enable peripheral 1 010: Enable peripheral 2 011: Enable peripheral 3 100: Enable peripheral 4 101: Enable peripheral 5 110: Enable peripheral 6 111: Enable peripheral 7 These bits can NOT be written when CHEN is '1'.
24:23	MBURST[1:0]	Transfer burst type of memory Software set and cleare. 00: single burst 01: INCR4 (4-beat incrementing burst) 10: INCR8 (8-beat incrementing burst) 11: INCR16 (16-beat incrementing burst) These bits can NOT be written when CHEN is '1'. 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 Software set and cleare. 00: single burst 01: INCR4 (4-beat incrementing burst) 10: INCR8 (8-beat incrementing burst) 11: INCR16 (16-beat incrementing burst) These bits can NOT be written when CHEN is '1'. 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 Hardware and software set, Hardware and software clear. 0: Memory 0 is selected as memory transfer area 1: Memory 1 is selected as memory transfer area This bit can NOT be written when CHEN is '1'. 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 Software set and clear. 0: Disable switch-buffer mode 1: Enable switch-buffer mode This bit can NOT be written when CHEN is '1'.
17:16	PRIO[1:0]	Priority level

		Software set and clear. 00: Low 01: Medium 10: High 11: Ultra high These bits can NOT be written when CHEN is '1'.
15	PAIF	Peripheral address increment fixed Software set and clear. 0: The peripheral address increment is determined by PWIDTH 1: The peripheral address increment is fixed to 4 This bit can NOT be written when CHEN is '1'. During the transmission, when PNAGA is configured to '0', this bit has no effect. 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 Software set and clear. 00: 8-bit 01: 16-bit 10: 32-bit 11: Reserved These bits can NOT be written when CHEN is '1'. 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 Software set and clear. 00: 8-bit 01: 16-bit 10: 32-bit 11: Reserved These bits can NOT be written when CHEN is '1'.
10	MNAGA	Next address generation algorithm of memory Software set and clear 0: Fixed address mode 1: Increasing address mode This bit can NOT be written when CHEN is '1'.
9	PNAGA	Next address generation algorithm of peripheral Software set and clear 0: Fixed address mode 1: Increasing address mode This bit can NOT be written when CHEN is '1'.

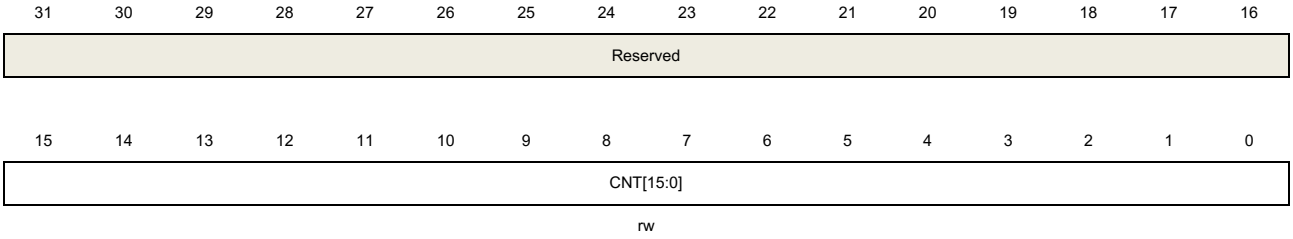
8	CMEN	Circular mode enable Software set and clear. 0: Disable circular mode. 1: Enable circular mode This bit can NOT be written when CHEN is '1'. This bit is automatically locked as '0' by hardware immediately after enable CHEN if TFCS is configured to '1'. This bit is automatically locked as '1' by hardware immediately after enable CHEN if SBMEN is configured to '1'.
7:6	TM[1:0]	Transfer mode Software set and clear. 00: Read from peripheral and write to memory 01: Read from memory and write to peripheral 10: Read from memory and write to memory 11: Reserved These bits can NOT be written when CHEN is '1'.
5	TFCS	Transfer flow controller select Software set and clear. 0: DMA is selected as the transfer flow controller 1: Peripheral is selected as the transfer flow controller This bit can NOT be written when CHEN is '1'.
4	FTFIE	Enable bit for full transfer finish interrupt Software set and clear. 0:Disable full transfer finish interrupt 1:Enable full transfer finish interrupt
3	HTFIE	Enable bit for half transfer finish interrupt Software set and clear. 0: Disable half transfer finish interrupt 1: Enable half transfer finish interrupt
2	TAEIE	Enable bit for tranfer access error interrupt Software set and clear. 0: Disable tranfer access error interrupt 1: Enable tranfer access error interrupt
1	SDEIE	Enable bit for single data mode exception interrupt Software set and clear. 0: Disable single data mode exception interrupt 1: Enable single data mode exception interrupt
0	CHEN	Channel enable Software set, hardware clear. 0: Disable channel 1: Enable channel

When this bit is asserted, the DMA transfer is started. This bit is automaticly cleared when one of the following situations occurs:
When the transfer of channel is fully finished.
When a wrong FIFO configuration or a transfer access error is detected.
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.

14.6.6. 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).



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CNT[15:0]	Transfer counter 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.

14.6.7. 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 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.

14.6.8. 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 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. 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'. 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. 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.

14.6.9. 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 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. 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'. 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. 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.

14.6.10. 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

		Software set and clear. 0: Disable FIFO error and exception interrupt 1: Enable FIFO error and exception interrupt
6	Reserved	Must be kept at reset value.
5:3	FCNT[2:0]	FIFO counter Hardware set and clear. 000: No data 001: One word 010: Two words 011: Three words 100: Empty 101: Full 110~111: Reserved These bits specific the number of data stored in FIFO during the transmission. When MDMEN is configured to '0', these bits has no meaning.
2	MDMEN	Multi-data mode enable Software set and clear. 0: Disable Multi-data mode 1: Enable Multi-data mode These bits can NOT be written when CHEN in the DMA_CHxCTL register is '1'. 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 Software set and clear 00: One word 01: Two Words 10: Three Words 11: Four Words These bits can NOT be written when CHEN in the DMA_CHxCTL register is '1'. When MDMEN is configured to '0', these bits has no meaning.

15. Image processing accelerator (IPA)

15.1. 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

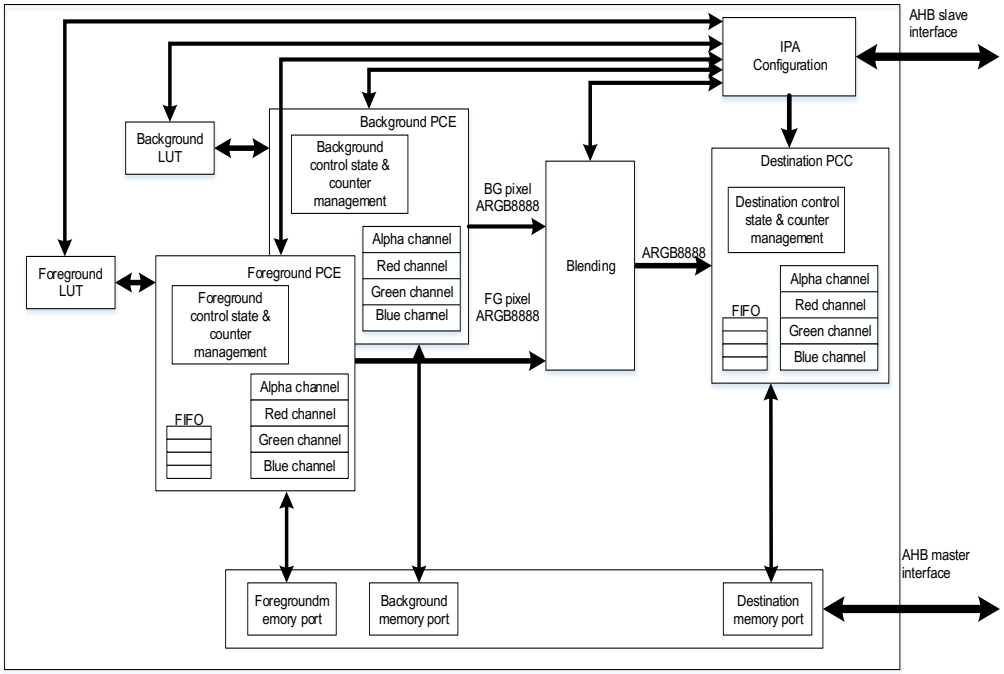
Eleven pixel formats from 4-bit up to 32-bit per pixel independently for the two source images and five pixel formats from 16-bit up to 32-bit per pixel for the destination image are supported. Two 256*32 bits LUTs (Look-Up Table) separately for the two source images are implemented for the indirect pixel formats.

15.2. Characteristics

- One AHB master interface for memory access and one AHB slave interface for IPA configuration with 8-bit, 16-bit and 32-bit
- Three four-word depth 32-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 eleven pixel formats independently for two source images
- Support five pixel formats for the destination image
- Support configurable image size
- Support automatically AHB bandwidth adjustment with an internal timer
- Support one interrupt with six types of event flags
- Support interrupt enable and clear

15.3. Block diagram

Figure 15-1. IPA block diagram



As showed in ***Figure 15-1. IPA block diagram***, the IPA consists of six main parts:

- IPA configuration through AHB slave interface
- Image data access through AHB master interface
- Foreground and background LUT
- Foreground and background pixel channel extension (PCE)
- Foreground and background pixel blending
- Destination pixel channel compression (PCC)

15.4. 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 15-1. 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 15-1. 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

15.4.1. 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 four-word depth 32-bit FIFOs are implemented for the foreground, background and 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 AHB 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.

15.4.2. 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 15-2. Foreground and background CLUT pixel format*.

Table 15-2. 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.

15.4.3. 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 15-3. 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.

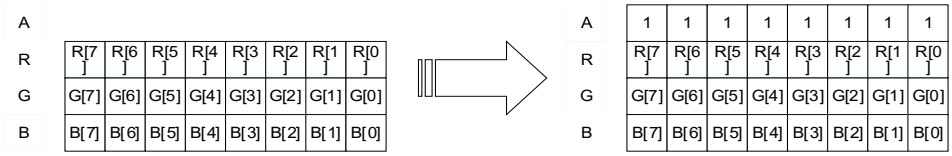
Table 15-3. Foreground and background pixel format

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]

If the pixel format is 'RGB888', the alpha channel value is equal to 0xFF when extending the pixel data, as shown in ***Figure 15-2. Pixel extension from 'RGB888' to 'ARGB8888'***.

Figure 15-2. Pixel extension from 'RGB888' to 'ARGB8888'

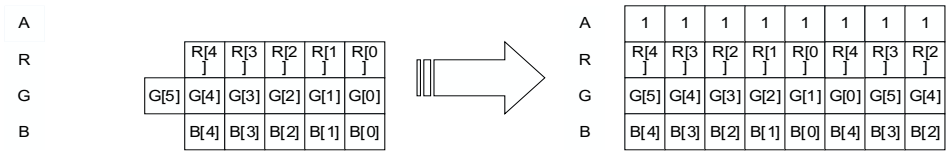
§ RGB888 à ARGB8888



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 15-3. Pixel extension from 'RGB565' to 'ARGB8888'***.

Figure 15-3. Pixel extension from 'RGB565' to 'ARGB8888'

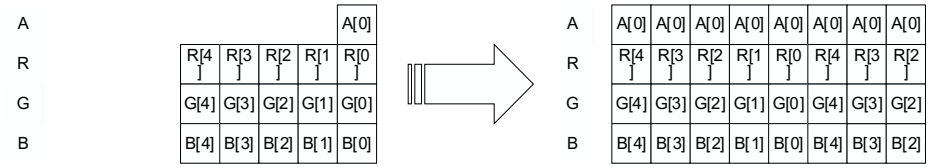
§ RGB565 à ARGB8888



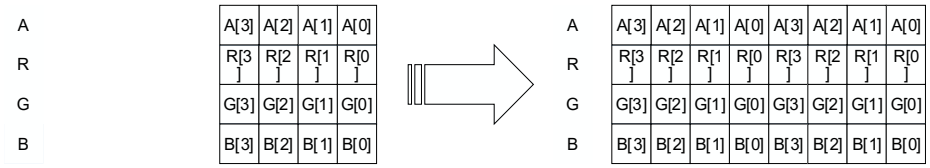
If the pixel format is 'ARGB1555' or 'ARGB4444', The value of every channel is extended to 8-bit by setting the MSBs to the original bits and copying the MSBs to the LSBs, as shown in the ***Figure 15-4. Pixel extension from 'ARGB1555' or 'ARGB4444' to 'ARGB8888'***.

Figure 15-4. Pixel extension from 'ARGB1555' or 'ARGB4444' to 'ARGB8888'

§ ARGB1555 à ARGB8888



§ ARGB4444 à 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.

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 15-4. Alpha channel value modulation***.

Table 15-4. 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

15.4.4. 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):

$$A_{mix} = \frac{A_F \times A_B}{255}$$

$$A_{blend} = A_F + A_B - A_{mix}$$

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):

$$R_{blend} = \frac{R_F \times A_F + R_B \times A_B - R_B \times A_{mix}}{A_{blend}}$$

$$G_{blend} = \frac{G_F \times A_F + G_B \times A_B - G_B \times A_{mix}}{A_{blend}}$$

$$B_{blend} = \frac{B_F \times A_F + B_B \times A_B - B_B \times A_{mix}}{A_{blend}}$$

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'.

15.4.5. 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 15-5. Destination pixel format*.

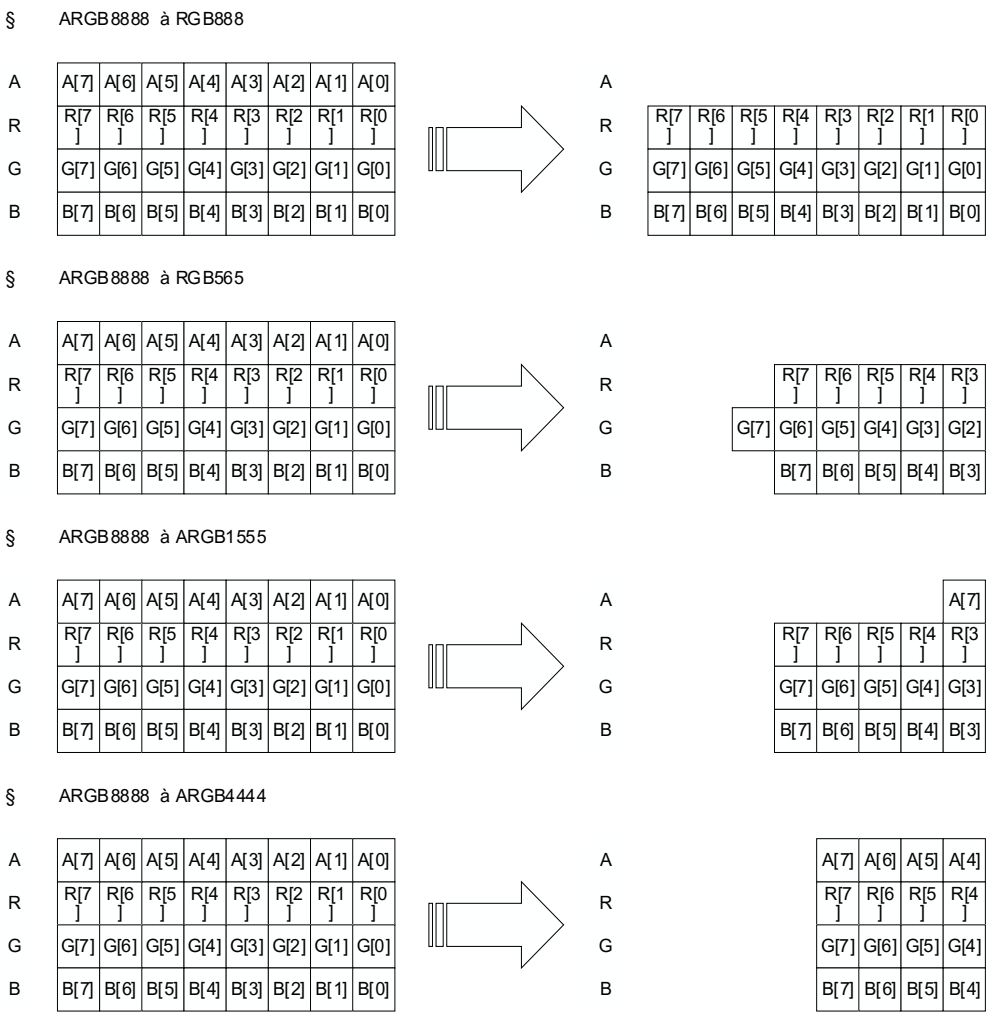
Table 15-5. Destination pixel format

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 15-5. Pixel compression*, the destination compression is performed by discarding the LSBs.

Figure 15-5. Pixel compression



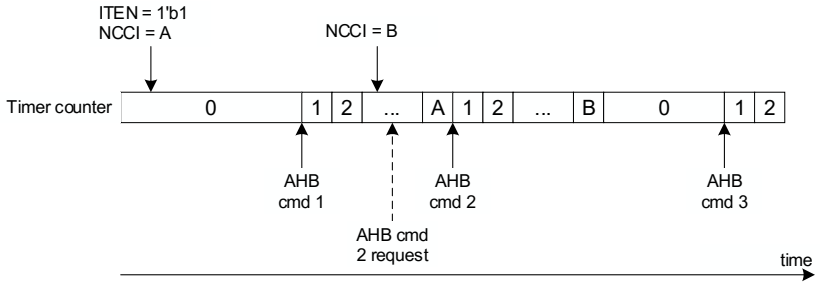
15.4.6. Inter-timer

To reduce the AHB bandwidth usage of IPA AHB master interface, a timer is implemented to insert a number of clocks between two consecutive AHB 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 AHB 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 15-6. Inter timer operation***.

Figure 15-6. Inter timer operation



15.4.7. 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.

15.4.8. 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.

15.4.9. 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.
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 LM bits in the IPA_LM register to set the line mark if needed.
7. Configure the WIDTH and HEIGHT bits in the IPA_IMS register to set the image size.
8. 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.
9. 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.
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.
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 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.

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' 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' or 'L4'.
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 '0b1010'.
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' or 'L4'.
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.

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) 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 12), 13) are considerable.

15.5. 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 15-6. IPA interrupt events*.

Table 15-6. 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.

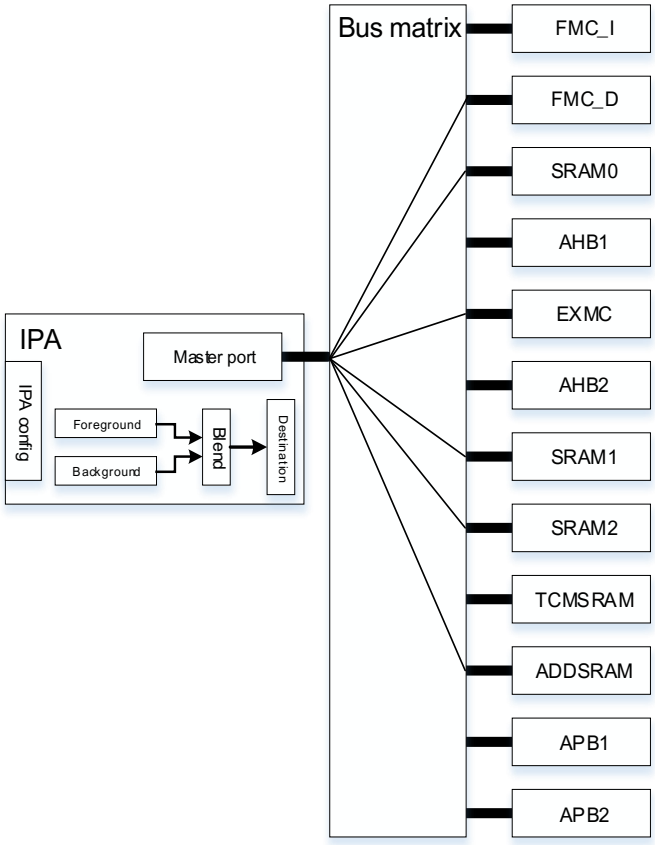
When the full transfer finish interrupt flag is asserted and the enabled bit for the full transfer finish interrupt is set, an IPA interrupt is generated.

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 15-7. 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 15-7. System connection of IPA



15.6. Register definition

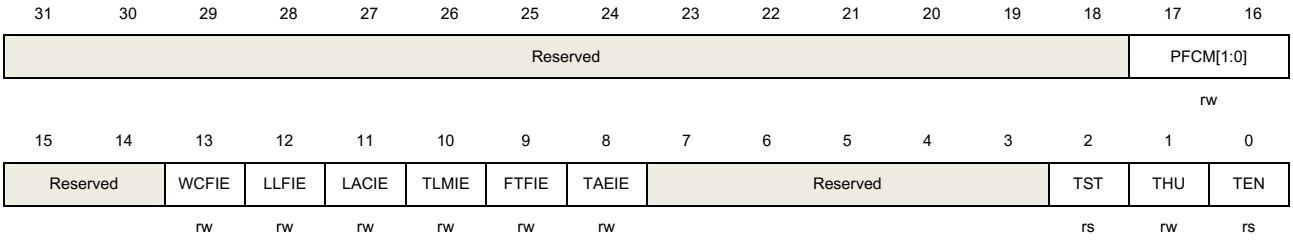
IPA base address: 0x4002 B000

15.6.1. 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 Software set and clear. 00: Foreground memory to destination memory without pixel format convert 01: Foreground memory to destination memory with pixel format convert 10: Blending foreground and background memory to destination memory 11: Fill up destination memory with specific color 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 Software set and clear 0: Disable configuration error interrupt 1: Enable configuration error interrupt
12	LLFIE	Enable bit for LUT loading finish interrupt Software set and clear 0: Disable LUT loading finish interrupt 1: Enable LUT loading finish interrupt
11	LACIE	Enable bit for LUT access conflict interrupt Software set and clear 0: Disable LUT access conflict interrupt 1: Enable LUT access conflict interrupt
10	TLMIE	Enable bit for transfer line mark interrupt Software set and clear

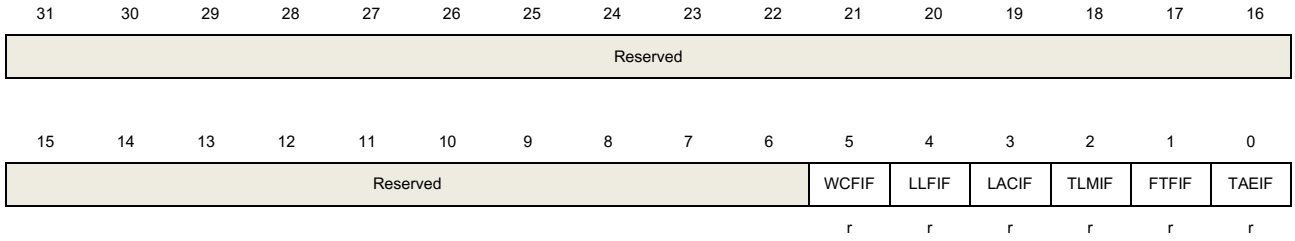
		0: Disable transfer line mark interrupt 1: Enable transfer line mark interrupt
9	FTFIE	Enable bit for full transfer finish interrupt Software set and clear 0: Disable full transfer finish interrupt 1: Enable full transfer finish interrupt
8	TAEIE	Enable bit for transfer access error interrupt Software set and clear 0: Disable transfer access error interrupt 1: Enable transfer access error interrupt
7:3	Reserved	Must be kept at reset value
2	TST	Transfer stop Software set, software and hardware clear. 0: No effect 1: Stop the current transfer 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 Software set, software and hardware clear. 0: No effect 1: Hang up the current transfer 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 Software set, hardware clear. 0: Transfer disable 1: Transfer enable When this bit is enabled, the IPA transfer is started. This bit is automatically cleared when one of the following situations occurs: – When the TST bit is enabled to stop the current transfer. – When the transfer is fully finished. – When a wrong configuration or a transfer access error is detected. – 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').

15.6.2. 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 Hardware set, Software cleared by enable 'WCFIFC' bit in the IPA_INTC register. 0: No wrong configuration is detected when IPA transfer or LUT loading is enable. 1: A wrong configuration is detected when IPA transfer or LUT loading is enable.
4	LLFIF	LUT loading finish interrupt flag Hardware set, software cleared by enable 'LLFIFC' bit in the IPA_INTC register. 0: No LUT loading finish is detected 1: A LUT loading finish is detected
3	LACIF	LUT access conflict interrupt flag Hardware set, software cleared by enable 'LACIFC' bit in the IPA_INTC register. 0: No LUT access conflict is detected. 1: A LUT access conflict is detected.
2	TLMIF	Transfer line mark interrupt flag Hardware set, Software cleared by enable 'CTCLIF' bit in the IPA_INTC register. 0: The number of pixel transferred has not exactly reached the line mark 1: The number of pixel transferred has exactly reached the line mark
1	FTFIF	Full transfer finish interrupt flag Hardware set, software cleared by enable 'CTFIF' bit in the IPA_INTC register. 0: No full transfer finish is detected. 1: A full transfer finish is detected.
0	TAEIF	Transfer access error interrupt flag Hardware set, software cleared by enable 'CTEIF' bit in the IPA_INTC register. 0: No transfer access error is detected. 1: A transfer access error is detected.

15.6.3. **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 Software set, hardware clear 0: No effect 1: Clear wrong configuration interrupt flag
4	LLFIFC	Clear bit for LUT loading finish interrupt flag Software set, hardware clear 0: No effect 1: Clear LUT loading finish interrupt flag
3	LACIFC	Clear bit for LUT access conflict interrupt flag Software set, hardware clear 0: No effect 1: Clear LUT access conflict interrupt flag
2	TLMIFC	Clear bit for transfer line mark interrupt flag Software set, hardware clear 0: No effect 1: Clear transfer line mark interrupt flag
1	FTFIFC	Clear bit for full transfer finish interrupt flag Software set, hardware clear 0: No effect 1: Clear full transfer finish interrupt flag
0	TAEIFC	Clear bit for transfer access error interrupt flag Software set, hardware clear 0: No effect 1: Clear transfer access error interrupt flag

15.6.4. 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 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, 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. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.5. 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													

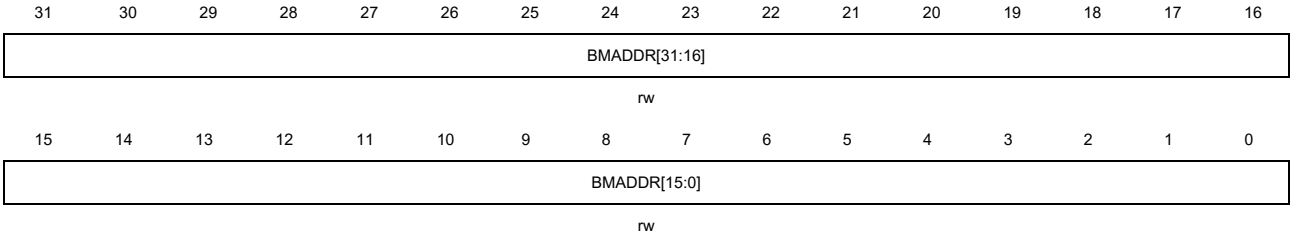
Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:0	FLOFF[13:0]	Foreground line offset These bits indicate the number of pixel between the last pixel of the current line and the first pixel of the next line. If the foreground pixel format is A4 or L4, the FLOFF must be configured to be an even number, otherwise a wrong configuration will be detected when the transfer is enable. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.6. 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).



Bits	Fields	Descriptions
31:0	BMADDR[31:0]	Background memory base address 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, 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. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.7. 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).



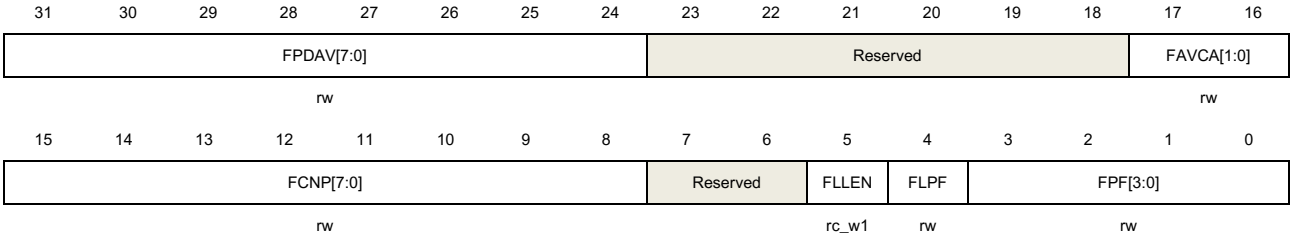
Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:0	BLOFF[13:0]	Background line offset These bits indicate the number of pixel between the last pixel of the current line and the first pixel of the next line. If the background pixel format is A4 or L4, the BLOFF must be configured to be an even number, otherwise a configuration error will be detected when the transfer is enable. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.8. 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 Software set and clear 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. 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	FAVCA[1:0]	Foreground alpha value calculation algorithm Software set and clear 00: No effect 01: FPDAV[7:0] is selected as the foreground alpha value 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 11: Reserved 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 Software set and clear The pixel number of foreground LUT is equal to FCNP + 1. 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 Software set, hardware clear. 0: Disable foreground LUT loading 1: Enable foreground LUT loading When this bit is enabled, the foreground LUT loading is started. This bit is automatically cleared when one of the following situations occurs: – When the TST bit is enabled – When the foreground LUT loading is finished – 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

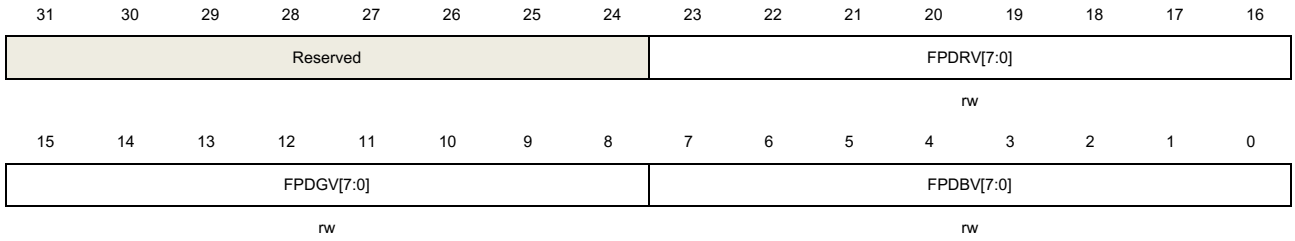
		Software set and clear 0: ARGB8888 1: RGB888 This bit can NOT be written when FLLEN is '1'.
3:0	FPF[3:0]	Foreground pixel format software set and clear 0000: ARGB8888 0001: RGB888 0010: RGB565 0011: ARGB1555 0100: ARGB4444 0101: L8 0110: AL44 0111: AL88 1000: L4 1001: A8 1010: A4 1011 ~ 1111: Reserved These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.9. 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 When the foreground pixel format is A4 or A8, these bits are used as the foreground red value. 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 Software set and clear When the foreground pixel format is A4 or A8, these bits are used as the

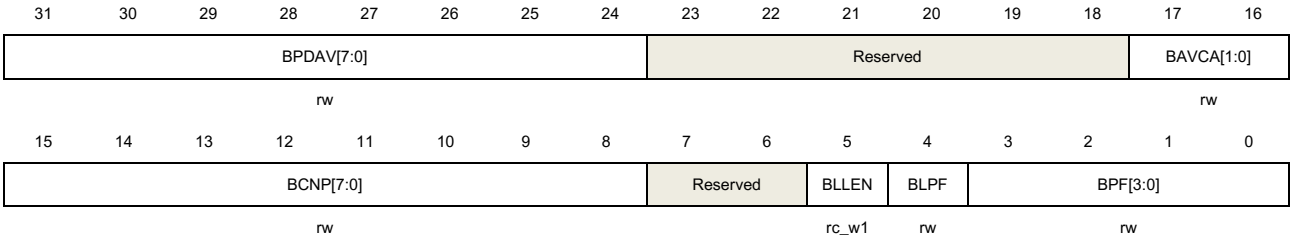
		foreground green value. These bits can NOT be written when TEN in the IPA_CTL register is '1'.
7:0	FPDBV[7:0]	Foreground pre-defined blue value Software set and clear When the foreground pixel format is A4 or A8, these bits are used as the foreground blue value. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.10. 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 Software set and clear 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. 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 Software set and clear 00: No effect 01: BPDAV[7:0] is selected as the foreground alpha value 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 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 Software set and clear The number of pixel of background LUT is equal to BCNP + 1. These bits can NOT be written when BLLEN is '1'.

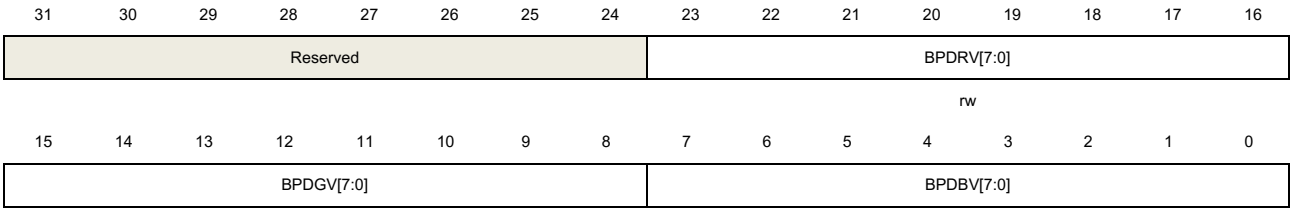
7:6	Reserved	Must be kept at reset value
5	BLLEN	Background LUT loading enable Software set, hardware clear. 0: Background LUT loading disable 1: Background LUT loading enable This bit is automatically cleared when one of the following situations occurs: – When the TST bit is enabled – When the background LUT loading is finished – When a wrong configuration or a transfer error is detected – 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 Software set and clear 0: ARGB8888 1: RGB888 This bit can NOT be written when BLLEN is '1'.
3:0	BPF[3:0]	Background pixel format software set and clear 0000: ARGB8888 0001: RGB888 0010: RGB565 0011: ARGB1555 0100: ARGB4444 0101: L8 0110: AL44 0111: AL88 1000: L4 1001: A8 1010: A4 1011 ~ 1111: Reserved These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.11. 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).



rw rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:16	BPDRV[7:0]	Background pre-defined red value Software set and clear When the background pixel format is A4 or A8, these bits are used as the background red value. 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 Software set and clear When the background pixel format is A4 or A8, these bits are used as the background green value. 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 Software set and clear When the background pixel format is A4 or A8, these bits are used as the background blue value. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.12. 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).

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

Bits	Fields	Descriptions
31:0	FLMADDR[31:0]	Foreground LUT memory base address 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'.

15.6.13. **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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLMADDR[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	BLMADDR[31:0]	Background LUT memory base address Software set and clear 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. These bit can NOT be written when BLLEN in the IPA_BPCTL register is '1'.

15.6.14. **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).

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													DPF[2:0]		
													rw		

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	DPF[2:0]	Destination pixel format Software set and clear 000: ARGB8888 001: RGB888 010: RGB565 011: ARGB1555 100: ARGB4444

101~111: Reserved

These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.15. **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 Software set and clear When IPA is configured to fill up destination memory with specific color, these bits are used as the destination alpha value. 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 Software set and clear 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'.
15:8	DPDGV[7:0]	Destination pre-defined green value Software set and clear 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 Software set and clear

When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.
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 Software set and clear 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'.
15:8	DPDGV[7:0]	Destination pre-defined green value Software set and clear 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 Software set and clear When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value. 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 Software set and clear 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'.
10:5	DPDGV[5:0]	Destination pre-defined green value Software set and clear 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'.
4:0	DPDBV[4:0]	Destination pre-defined blue value Software set and clear

When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value.
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 Software set and clear When IPA is configured to fill up destination memory with specific color, these bits are used as the destination alpha value. 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 Software set and clear 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 Software set and clear 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'.
4:0	DPDBV[4:0]	Destination pre-defined blue value Software set and clear When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value. 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 Software set and clear When IPA is configured to fill up destination memory with specific color, these bits are used as the destination alpha value. 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 Software set and clear

		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'.
7:4	DPDGV[3:0]	Destination pre-defined green value Software set and clear 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'.
3:0	DPDBV[3:0]	Destination pre-defined blue value Software set and clear When IPA is configured to fill up destination memory with specific color, these bits are used as the destination blue value. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.16. 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 software set and clear 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. These bit can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.17. 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 software set and clear 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. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.18. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		WIDTH[13:0]													
		rw													

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

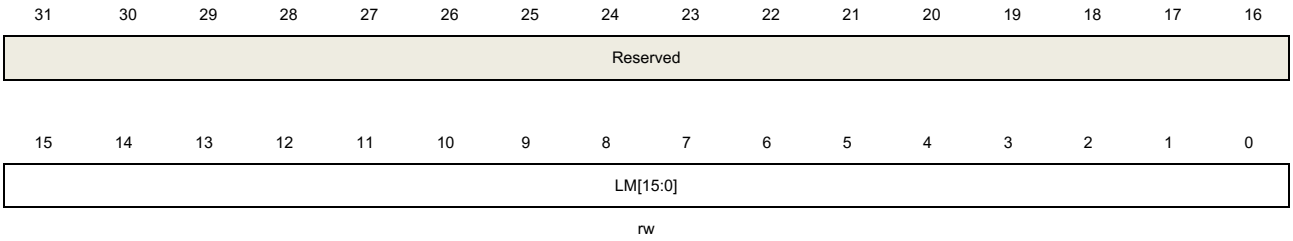
Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:16	WIDTH[13:0]	Width of the image to be processed Software set and clear These bits signify the number of pixels per line. If the foreground or background pixel format is A4 or L4, these bits must be configured to be an even number, otherwise a wrong configuration will be detected when the transfer is enable. 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 Software set and clear These bits specify the number of lines of image to be processed. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.19. 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).



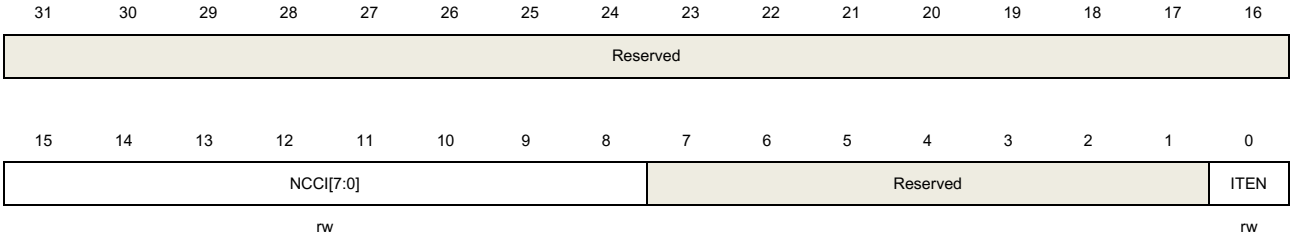
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	LM[15:0]	line mark Software set and clear These bits define a line number to signify the transfer level. An interrupt flag is asserted as soon as the last pixel of the marked line has been written into the destination memory. These bits can NOT be written when TEN in the IPA_CTL register is '1'.

15.6.20. 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 Software set and clear 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

An inter-timer is implemented to reduce the AHB bus bandwidth usage of IPA.

0: Disable inter-timer

1: Enable inter-timer

16. Debug (DBG)

16.1. Overview

The GD32F527xx 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®-M33. 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®-M33 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, RTC, I2C or CAN.

16.2. 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).

16.3. Switch JTAG or SW interface

By default, the JTAG interface is active. The sequence for switching from JTAG to SWD is:

- Send 50 or more TCK cycles with TMS = 1
- Send the 16-bit sequence on TMS = 1110011110011110 (0xE79E LSB first)
- Send 50 or more TCK cycles with TMS = 1

The sequence for switching from SWD to JTAG is:

- Send 50 or more TCK cycles with TMS = 1
- Send the 16-bit sequence on TMS = 1110011100111100 (0xE73C LSB first)
- Send 50 or more TCK cycles with TMS = 1

16.3.1. 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 the async trace is enabled.

Table 16-1. 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/PB4/PB3 are released to other GPIO functions. If JTAG and SW not used, all 5-pin can be released to other GPIO functions. Please refer to ***General-purpose and alternate-function I/Os (GPIO and AFIO)*** .

16.3.2. JTAG daisy chained structure

The Cortex®-M33 JTAG TAP is sconnected to a MCU JTAG TAP. The MCU JTAG IR is 5-bit width, while the Cortec-M33 JTAG IR is 4-bit width. So when JTAG in IR shift step, it first shift 5-bit BYPASS instruction (5'b 11111) for MCU JTAG, and then shift normal 4-bit instruction for Cortex®-M33 JTAG. Because of the data shift under MCU JTAG BYPASS mode, adding 1 extra bit to the data chain is needed.

The MCU JTAG IDCODE is 0x790007A3.

16.3.3. Debug reset

The JTAG-DP and SW-DP register are in the power on reset domain. The System reset initializes the majority of the Cortex®-M33, 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.

16.3.4. JEDEC-106 ID code

The Cortex®-M33 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00FF000_0xE00FFFFF.

16.4. Debug hold function overview

16.4.1. Debug support for power saving mode

When STB_HOLD bit in DBG control register 0 (DBG_CTL0) is set and entering the standby mode, the clock of AHB bus and system clock are provided by CK_IRC16M, 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 0 (DBG_CTL0) is set and entering the Deep-

sleep mode, the clock of AHB bus and system clock are provided by CK_IRC16M, and the debugger can debug in Deep-sleep mode.

When SLP_HOLD bit in DBG control register 0 (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.

16.4.2. Debug support for TIMER, I2C, RTC, WWDGT, FWDGT and CAN

When the core halted and the corresponding bit in DBG control register 1 (DBG_CTL1) or DBG control register 2 (DBG_CTL2) 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 stopped for debug.

For CAN, the receive register stopped counting for debug.

16.5. Register definition

DBG base address: 0xE0044000

16.5.1. 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 These bits read by software, These bits are unchanged constant.

16.5.2. Control register 0 (DBG_CTL0)

Address offset: 0x04
Reset value: 0x0000 0000; power reset 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TRACE_IOEN	Reserved		STB_HOLD	DSLP_HOLD	SLP_HOLD
										rw			rw	rw	rw

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	TRACE_IOEN	Trace pin allocation enable This bit is set and reset by software. 0: Trace pin allocation disable 1: Trace pin allocation enable
4:3	Reserved	Must be kept at reset value.
2	STB_HOLD	Standby mode hold bit This bit is set and reset by software

0: no effect
1: At the standby mode, the clock of AHB bus and system clock are provided by CK_IRC16M, a system reset generated when exit standby mode.

1 DSLP_HOLD Deep-sleep mode hold bit
This bit is set and reset by software
0: no effect
1: At the Deep-sleep mode, the clock of AHB bus and system clock are provided by CK_IRC16M.

0 SLP_HOLD Sleep mode hold bit
This bit is set and reset by software
0: no effect
1: At the sleep mode, the clock of AHB is on.

16.5.3. Control register 1 (DBG_CTL1)

Address offset: 0x08
Reset value: 0x0000 0000; power reset 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					CAN1_HOLD	CAN0_HOLD	Reserved	I2C2_HOLD	I2C1_HOLD	I2C0_HOLD	I2C5_HOLD	I2C4_HOLD	I2C3_HOLD	Reserved	
					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			FWDGT_HOLD	WWDGT_HOLD	RTC_HOLD	Reserved	TIMER13_HOLD	TIMER12_HOLD	TIMER11_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
31:27	Reserved	Must be kept at reset value.
26	CAN1_HOLD	CAN1 hold bit This bit is set and reset by software 0: no effect 1: the receive register of CAN1 stops receiving data when core halted.
25	CAN0_HOLD	CAN0 hold bit This bit is set and reset by software 0: no effect 1: the receive register of CAN0 stops receiving data when core halted.
24	Reserved	Must be kept at reset value.
23	I2C2_HOLD	I2C2 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C2 SMBUS timeout for debug when core halted.

22	I2C1_HOLD	I2C1 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C1 SMBUS timeout for debug when core halted.
21	I2C0_HOLD	I2C0 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C0 SMBUS timeout for debug when core halted.
20	I2C5_HOLD	I2C5 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C5 SMBUS timeout for debug when core halted.
19	I2C4_HOLD	I2C4 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C4 SMBUS timeout for debug when core halted.
18	I2C3_HOLD	I2C3 hold bit This bit is set and reset by software 0: no effect 1: hold the I2C3 SMBUS timeout for debug when core halted.
17:13	Reserved	Must be kept at reset value.
12	FWDGT_HOLD	FWDGT hold bit This bit is set and reset by software 0: no effect 1: hold the FWDGT counter clock for debug when core halted.
11	WWDGT_HOLD	WWDGT hold bit This bit is set and reset by software 0: no effect 1: hold the WWDGT counter clock for debug when core halted.
10	RTC_HOLD	RTC hold bit This bit is set and reset by software 0: no effect 1: hold the RTC counter for debug when core halted.
9	Reserved	Must be kept at reset value.
8	TIMER13_HOLD	TIMER 13 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 13 counter for debug when core halted.
7	TIMER12_HOLD	TIMER 12 hold bit This bit is set and reset by software

0: no effect
1: hold the TIMER 12 counter for debug when core halted.

6	TIMER11_HOLD	TIMER 11 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 11 counter for debug when core halted.
5	TIMER6_HOLD	TIMER 6 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 6 counter for debug when core halted.
4	TIMER5_HOLD	TIMER 5 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 5 counter for debug when core halted.
3	TIMER4_HOLD	TIMER 4 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 4 counter for debug when core halted.
2	TIMER3_HOLD	TIMER 3 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 3 counter for debug when core halted.
1	TIMER2_HOLD	TIMER 2 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 2 counter for debug when core halted.
0	TIMER1_HOLD	TIMER 1 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 1 counter for debug when core halted.

16.5.4. Control register 2 (DBG_CTL2)

Address offset: 0x0C
Reset value: 0x0000 0000; power reset 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													TIMER10_HOLD	TIMER9_HOLD	TIMER8_HOLD
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	TIMER7_HOLD	TIMER0_HOLD
	rw	rw

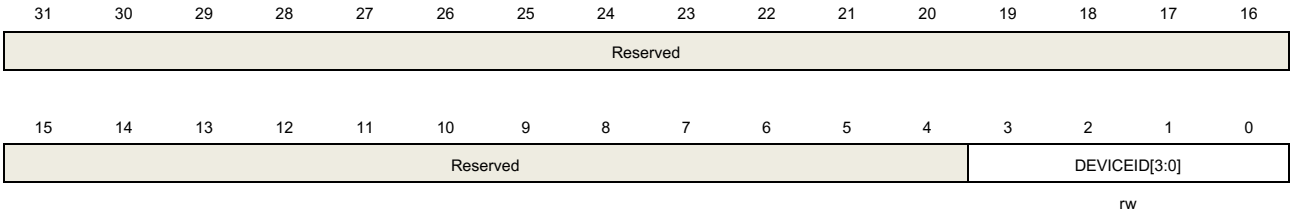
Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value.
18	TIMER10_HOLD	TIMER 10 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 10 counter for debug when core halted.
17	TIMER9_HOLD	TIMER 9 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 9 counter for debug when core halted.
16	TIMER8_HOLD	TIMER 8 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 8 counter for debug when core halted.
15:2	Reserved	Must be kept at reset value.
1	TIMER7_HOLD	TIMER 7 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 7 counter for debug when core halted.
0	TIMER0_HOLD	TIMER 0 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 0 counter for debug when core halted.

16.5.5. Control register 3 (DBG_CTL3)

Address offset: 0x10

Reset value: 0x0000 0000; power reset only

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



Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3:0	DEVICEID[3:0]	deviceid which connect to CPU instanceid These bits can be written and read by software

17. Programmable current reference (IREF)

17.1. Overview

A programmable current reference module is included in the MCU.

Two different running modes are supplied for user to use current reference, one mode named Low Power Mode (LPM) and another one named High Current Mode (HCM).

The difference between two modes is the current step and maximum current.

The former's (LPM) step current is 1uA and the (HCM) 8uA.

The former's (LPM) maximum current is 63uA and the (HCM) 504uA.

17.2. Characteristics

Current reference features:

- Programmable current.
- Programmable source or sink.
- Low Power Mode(LPM) and High Current Mode(HCM).

17.3. Function overview

17.3.1. Signal description

When IREF is used, the relevant pin should be configured to analog input mode.

17.3.2. User trimming

User can trim the IREF output current by programming CPT bit in IREF_CTL register.

17.4. Register definition

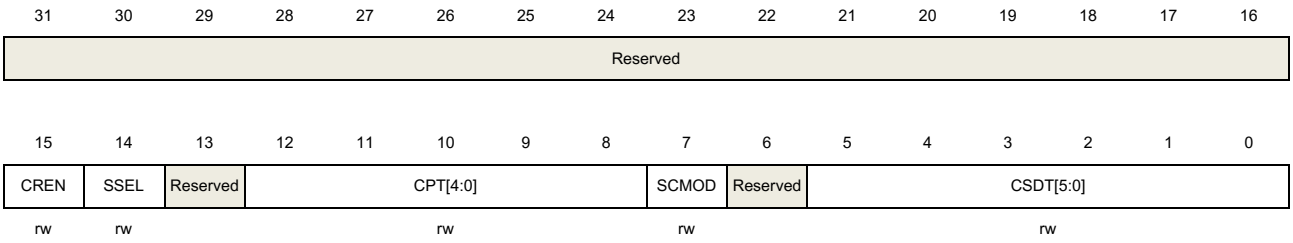
IREF base address: 0x4000 C400

17.4.1. Control register (IREF_CTL)

Address offset: 0x300

Reset value: 0x0000 0F00

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



timer	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	CREN	Current reference enable 0: Disable current reference 1: Enable current reference
14	SSEL	Step selection 0: Low power, 1uA step 1: High current, 8uA step
13	Reserved	Must be kept at reset value
12:8	CPT[4:0]	Current precision trim for output current offset 0x00: -30% …. 0x1F: +32%
7	SCMOD	Sink current mode 0: Source current mode 1: Sink current mode
6	Reserved	Must be kept at reset value.
5:0	CSDT[5:0]	Current step data 0x00: Default value. 0x01: Step * 1 …. 0x3F: Step * 63

18. Analog-to-digital converter (ADC)

18.1. Overview

A 12-bit successive approximation analog-to-digital converter module(ADC) is integrated on the MCU chip, which can sample analog signals from 16 external channels and 2 internal channels and the battery voltage (V_{BAT}) channel. The 19 ADC sampling channels all support a variety of operation modes. 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. An on-chip hardware oversample scheme improves performances and reduces the computational burden of MCU. For motors, power supplies and other applications that have a higher demand for ADC, you can contact our sales staff for more ADC details.

18.2. Characteristics

- High performance.
  - ADC sampling rsolution: 12-bit, 10-bit, 8-bit or 6-bit.
  - ADC sampling rate: 2.6 MSPs for 12-bit resolution, 3.0 MSPs for 10-bit resolution, faster sampling rate can be obtained by lowering the resolution.
  - Self-calibration time: 131 ADC clock periods.
  - Programmable sampling time.
  - Data storage mode: the most significant bit and the least significant bit.
  - DMA support.
- Analog input channels.
  - 16 external analog inputs.
  - 1 channel for internal temperature sensor (V_{SENSE}).
  - 1 channel for internal reference voltage (V_{REFINT}).
  - 1 channel for external battery power supply pin (V_{BAT}).
- Start-of-conversion can be initiated.
  - By software.
  - By hardware triggers.
- 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 or more ADCs).
- Conversion result threshold monitor function: analog watchdog.
- Interrupt generation.
  - At the end of routine conversions.
  - Analog watchdog event.

  - Overflow event.
- Oversampler.
  - 16-bit data register.
  - Oversampling ratio adjustable from 2x to 256x.
  - Programmable data shift up to 8-bit.
- Module supply requirements: 1.8V to 3.6V, and typical power supply voltage is 3.3V.
- Channel input range: V_{REF-} <=V_{IN} <=V_{REF+}.

18.3. Pins and internal signals

Figure 18-1. ADC module block diagram shows the ADC block diagram, *Table 18-1. ADC internal input signals* gives the ADC internal signals and *Table 18-2. ADC input pins definition* gives the ADC pin description.

Table 18-1. ADC internal input signals

Internal signal name	Description
V_{SENSE}	Internal temperature sensor output voltage
V_{REFINT}	Internal voltage reference output voltage

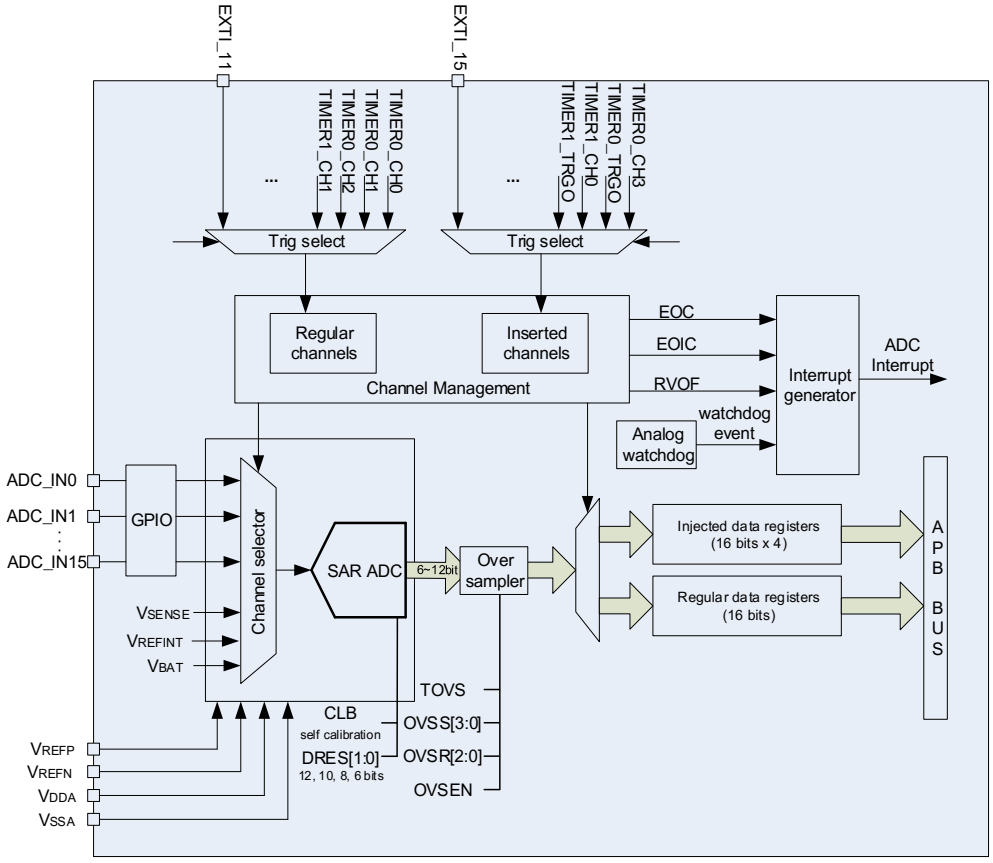
Table 18-2. ADC input pins definition

Name	Description
V_{DDA}	Analog power supply equal to V_{DD} and 1.8 V <= V_{DDA}<= 3.6 V
V_{SSA}	Ground for analog power supply equal to V_{SS}
V_{REF+}	The positive reference voltage for the ADC, 1.8 V <=V_{REF+}<= V_{DDA}
V_{REF-}	The negative reference voltage for the ADC,V_{REF-} = V_{SSA}
ADCx_IN[15:0]	Up to 16 external channels
V_{BAT}	External battery voltage

Note: V_{DDA} and V_{SSA} have to be connected to V_{DD} and V_{SS}, respectively.

18.4. Function overview

Figure 18-1. ADC module block diagram



18.4.1. 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 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.

When the ADC operating conditions change (such as supply power voltage VDDA, positive reference voltage VREFP, 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

18.4.2. ADC clock

The CK_ADC clock is synchronous with the AHB and APB2 clock and provided by the clock controller. The maximum frequency is 40MHz. The ADC clock can be allocated and configured by the ADCCK[2:0] bits in the ADC_SYNCCTL register. The ADC_SYNCCTL register is only valid for ADC0. Therefore, to configure the clock division for ADC1 and ADC2, the clock for ADC0 must be enabled.

18.4.3. ADCON 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 submodule will be put into powerdown mode. After ADC is enabled, you need delay t_{SU} time for sampling, the value of t_{SU} please refer to the device datasheet.

18.4.4. Routine sequence

The channel management circuit can organize the sampling conversion channels into a sequence: routine sequence. The routine sequence supports up to 16 channels, and each channel is called routine channel.

The RL[3:0] bits in the ADC_RSQ0 register specify the total conversion sequence length. The ADC_RSQ0~ADC_RSQ2 registers specify the selected channels of the routine sequence.

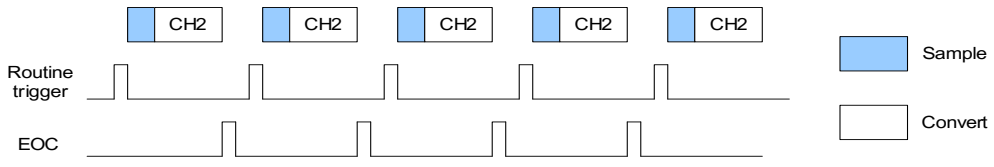
Note: Although the ADC supports 19 multiplexed channels, the maximum length of the sequence is only 16.

18.4.5. 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_RSQ2 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 external trigger is active.

Figure 18-2. 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.

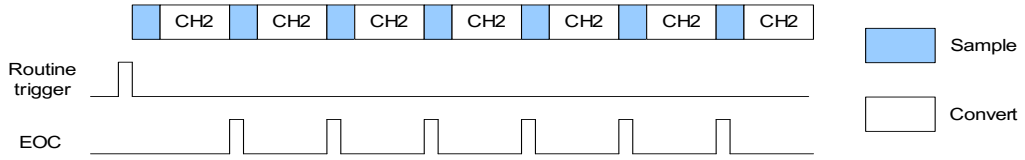
Software procedure for single operation mode of a routine channel:

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_SAMPTx register
4. Configure ETMRC and ETSRC bits in the ADC_CTL1 register if in need
5. Set the SWRCST bit, or generate an external 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[4:0]. When the ADCON has been set high, the ADC samples and converts specified channel, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC_RDATA register.

Figure 18-3. Continuous operation mode



Software procedure for continuous operation 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_SAMPTx register
4. Configure ETMRC and ETSRC bits in the ADC_CTL1 register if in need
5. Set the SWRCST bit, or generate an external 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_SAMPTx register

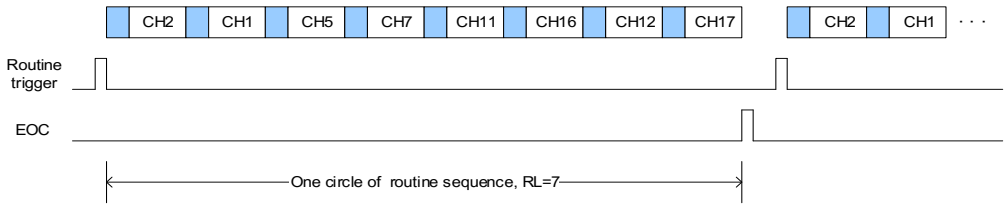
4. Configure ETMRC and ETSRC bits in the ADC_CTL1 register if in need
5. Prepare the ***Direct memory access controller (DMA)*** module to transfer data from the ADC_RDATA.
6. Set the SWRCST bit, or generate an external trigger for the routine sequence

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_RSQ2 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 external 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.

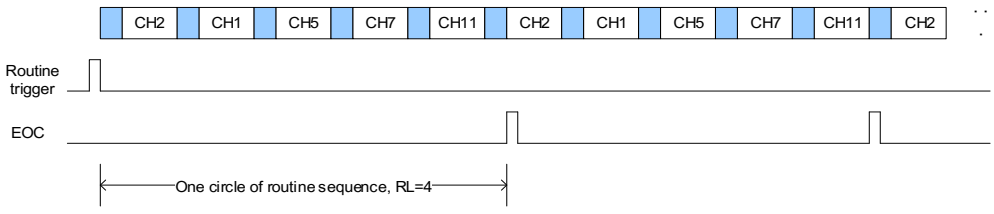
Figure 18-4. Scan operation mode, continuous disable



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 and ADC_SAMPTx registers
3. Configure ETMRC and ETSRC bits in the ADC_CTL1 register if in need
4. Prepare the ***Direct memory access controller (DMA)*** module to transfer data from the ADC_RDATA.
5. Set the SWRCST bit, or generate an external trigger for the routine sequence

Figure 18-5. Scan operation mode, continuous enable

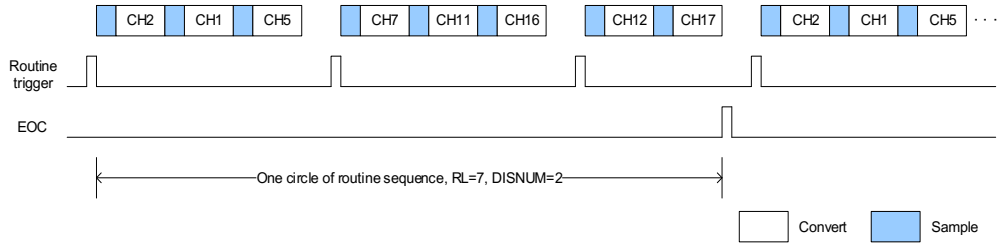


Discontinuous operation mode

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 conversions selected in the ADC_RSQ0~ADC_RSQ2 registers. The value of n is configured by the DISNUM[2:0] bits in the ADC_CTL0 register. When the corresponding software trigger or external trigger is active, the ADC samples and converts the next n channels configured in the ADC_RSQ0~ADC_RSQ2 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 18-6. 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 and ADC_SAMPTx registers
4. Configure ETMRC and ETSRC bits in the ADC_CTL1 register if in need
5. Prepare the ***Direct memory access controller (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 external trigger for the routine sequence
7. Repeat step6 if in need.
8. Wait the EOC flag to be set
9. Clear the EOC flag by writing 0 to it

18.4.6. Conversion result threshold monitor function

The analog watchdog is enabled when the RWDEN 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 the WDE bit in ADC_STAT register will be set. An interrupt will be generated if the WDEIE bit is set. The ADC_WDHT and ADC_WDLT registers are used to specify the high and low threshold. The comparison is done before the alignment, so the threshold value is 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 RWDEN, WDSC and WDCHSEL[4:0] bits in ADC_CTL0 register, can be monitored by the analog watchdog.

18.4.7. Data storage mode

The alignment of data stored after conversion can be specified by DAL bit in the ADC_CTL1 register.

Figure 18-7. Data storage mode of 12-bit resolution

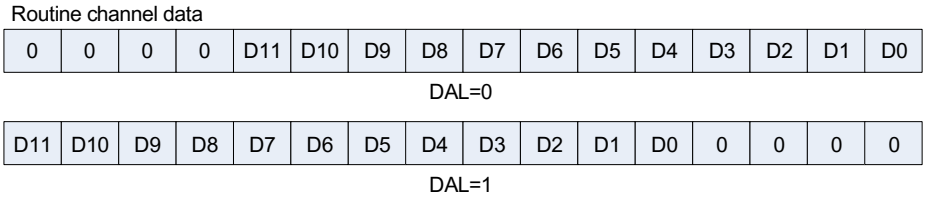


Figure 18-8. Data storage mode of 10-bit resolution

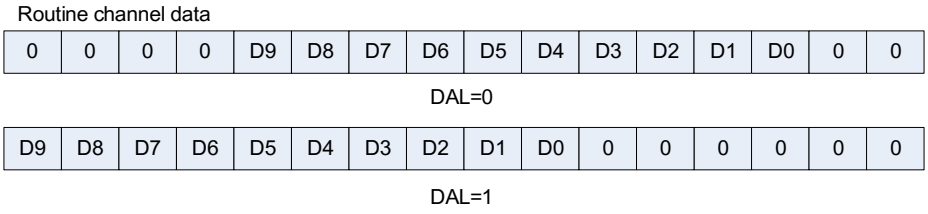
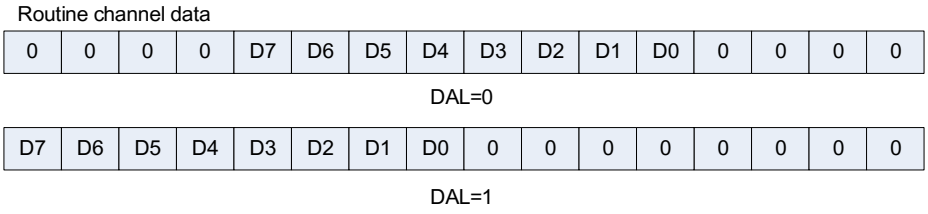
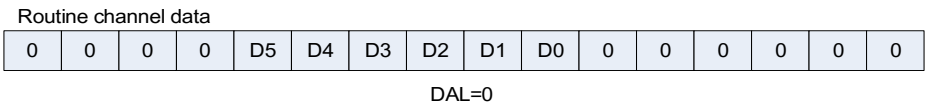


Figure 18-9. Data storage mode of 8-bit resolution



6-bit resolution data storage mode is different from 12-bit/10-bit/8-bit resolution data storage mode, shown as ***Figure 18-10. Data storage mode of 8-bit resolution***.

Figure 18-10. Data storage mode of 8-bit resolution



18.4.8. Sample time configuration

The number of CK_ADC cycles which is used to sample the input voltage can be specified by the SPTn[2:0] bits in the ADC_SAMPT0 and ADC_SAMPT1 registers. A different sample time can be specified for each channel. For 12-bits resolution, the total sampling and conversion time is "sampling time + 12" CK_ADC cycles.

Example:

CK_ADC = 40MHz and sample time is 3 cycles, the total conversion time is "3+12" CK_ADC cycles, that means 0.375us.

18.4.9. **External trigger** configuration

The conversion of routine sequence can be triggered by rising/falling edge of external trigger inputs. The ETMRC[1:0] bits in the ADC_CTL1 register control the trigger modes of r routine sequence. The external trigger source of routine sequence is controlled by the ETSRC[3:0] bits in the ADC_CTL1 register.

Table 18-3. External trigger modes

ETMRC[1:0]	Trigger mode
00	External trigger disable
01	Rising edge of external trigger enable
10	Falling edge of external trigger enable
11	Rising and falling edge of external trigger enable

Table 18-4. External trigger source for ADC

ETSRC[3:0]	Trigger Source	Trigger Type
0000	TIMER0_CH0	Hardware trigger
0001	TIMER0_CH1	
0010	TIMER0_CH2	
0011	TIMER1_CH1	
0100	TIMER1_CH2	
0101	TIMER1_CH3	
0110	TIMER1_TRGO	
0111	TIMER2_CH0	
1000	TIMER2_TRGO	
1001	TIMER3_CH3	
1010	TIMER4_CH0	
1011	TIMER4_CH1	
1100	TIMER4_CH2	
1101	TIMER7_CH0	
1110	TIMER7_TRGO	
1111	EXTI_11	

The selection of the external triggers can be changed on the fly, while no trigger event occurs due to this change.

18.4.10. **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.

18.4.11. 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 ADON 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_{SU}
9. Start conversion with software or trigger.

18.4.12. ADC internal channels

When the TSVREN bit of ADC_SYNCCTL register is set, the temperature sensor channel (ADC0_IN16) and V_{REFINT} channel (ADC0_IN17) 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 TSVREN bit.

The output voltage of the 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 process 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 ADC0_IN17 input channel.

To use the temperature sensor:

1. Configure the conversion sequence (ADC0_IN16) and the sampling time (t_{s_temp} μs) for the channel.
2. Enable the temperature sensor by setting the TSVREN 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 internal temperature sensor output voltage($V_{temperature}$), 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 please refer to the datasheet.
Avg_Slope: average slope for curve between temperature vs. internal temperature sensor output voltage, the typical value please refer to the datasheet.

18.4.13. 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 of ADC_SYNCCTL register is set, V_{BAT} channel (ADC_IN18) 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 ADC_IN18 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.

18.4.14. Programmable resolution (DRES)

The resolution is configured 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. The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeroes. Lower resolution reduces the conversion time needed for the successive approximation steps as shown in *Table 18-5. tCONV timings depending on resolution* .

Table 18-5. t_{CONV} timings depending on resolution

DRES[1:0] bits	t_{CONV} (ADC clock cycles)	t_{CONV}(ns) at f_{ADC}=40MHz	t_{SMPL}(min) (ADC clock cycles)	t_{ADC} (ADC clock cycles)	t_{ADC}(us) at f_{ADC}=40MHz
12	12	300 ns	3	15	375 ns
10	10	250 ns	3	13	325 ns
8	8	200 ns	3	11	275 ns
6	6	150 ns	3	9	225 ns

18.4.15. On-chip hardware oversampling

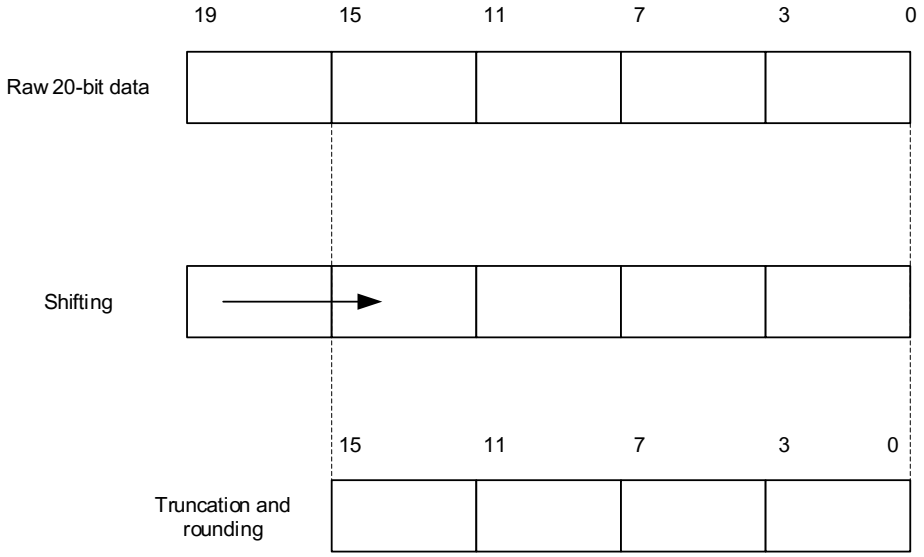
The on-chip hardware oversampling circuit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit. It provides a result with the following form, where N and M can be adjusted, and $D_{out}(n)$ is the n-th output digital signal of the ADC:

$$Result = \frac{1}{M} * \sum_{n=0}^{N-1} D_{out}(n) \qquad (18\text{-}1)$$

The on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting. The oversampling ratio N is defined by the OVSR[2:0] 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.

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.

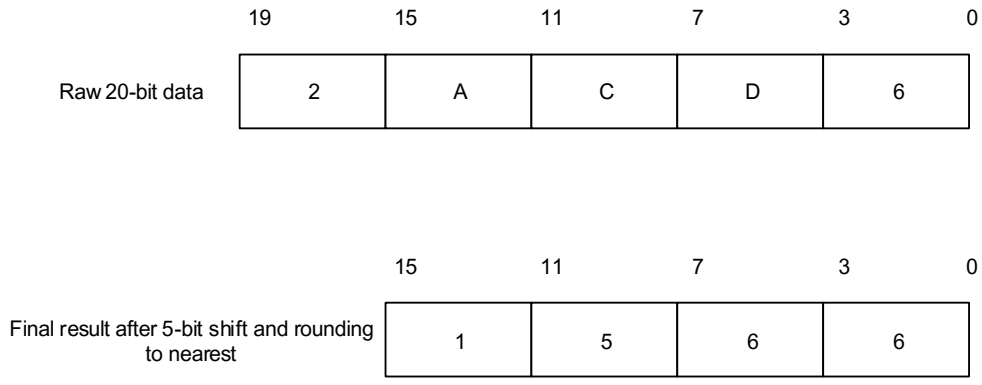
Figure 18-11. 20-bit to 16-bit result truncation



Note: If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

Figure 18-11. 20-bit to 16-bit result truncation shows a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 18-12. Numerical example with 5-bits shift and rounding



The *Table 18-6. Maximum output results vs N and M Grayed values indicates*

truncation gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFF.

Table 18-6. Maximum output results vs N and M Grayed values indicates truncation

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

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sampling time remains equal throughout the oversampling sequence. New data is supplied every N conversions, and the equivalent delay is equal to:

$$N \times t_{ADC} = N \times (t_{SMPL} + t_{CONV}) \quad (18\text{-}2)$$

18.5. ADC sync mode

In devices with more than one ADC, the ADC sync mode can be used. In ADC sync mode, the conversion starts alternately or simultaneously triggered by ADC0/ADC1/ADC2, according to the sync mode configurated by the SYNCM[4:0] 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 and ADC2. The converted data of routine channel is stored in the ADC sync routine data register (ADC_SYNCDATA).

The following modes can be configured in *Table 18-7. ADC sync mode table*.

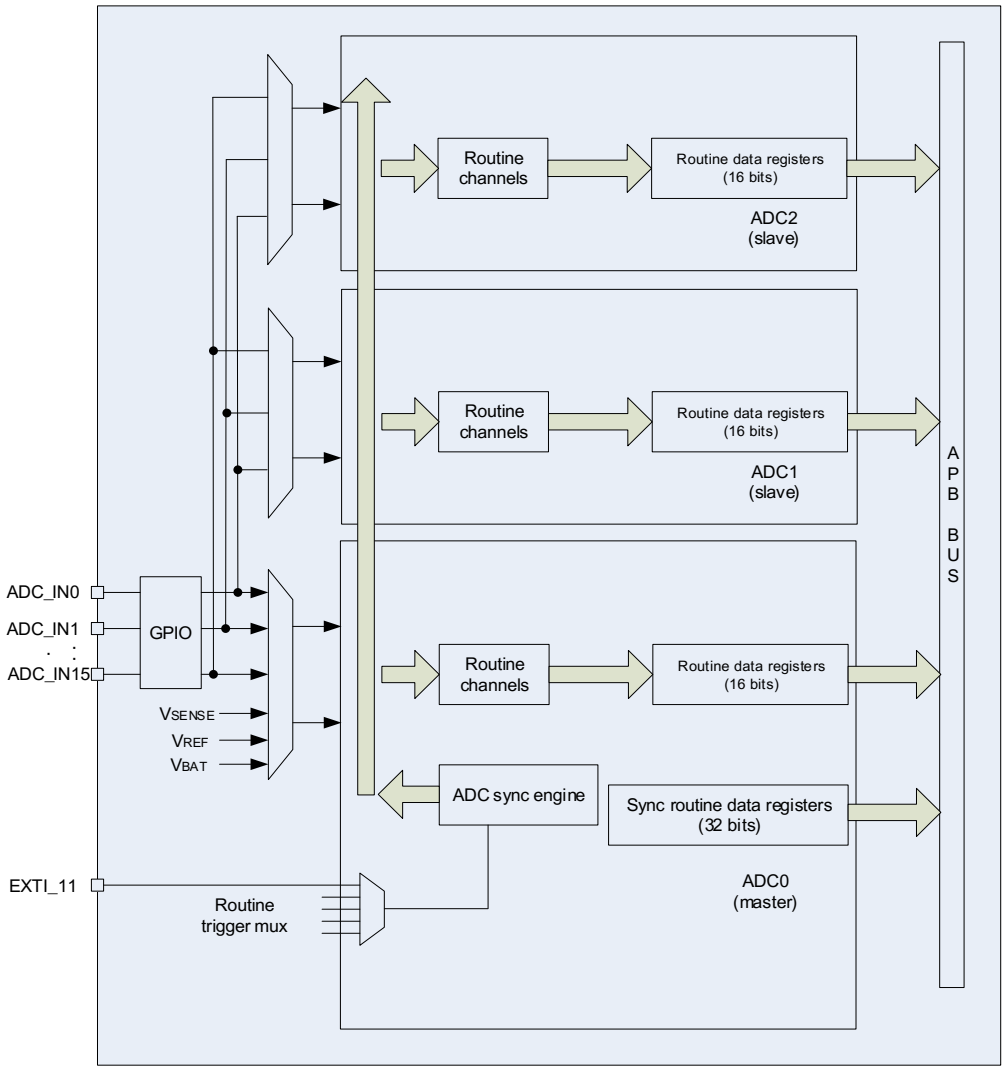
Table 18-7. ADC sync mode table

SYNCM[4: 0]	mode
00000	Free mode
00110	ADC0 and ADC1 work in routine parallel mode
00111	ADC0 and ADC1 work in routine follow-up mode
10110	All ADCs work in routine parallel mode
10111	All ADCs work in 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 18-13. ADC sync block diagram* .

Figure 18-13. ADC sync block diagram



18.5.1. Free mode

In this mode, each ADC works independently and does not interfere with each other.

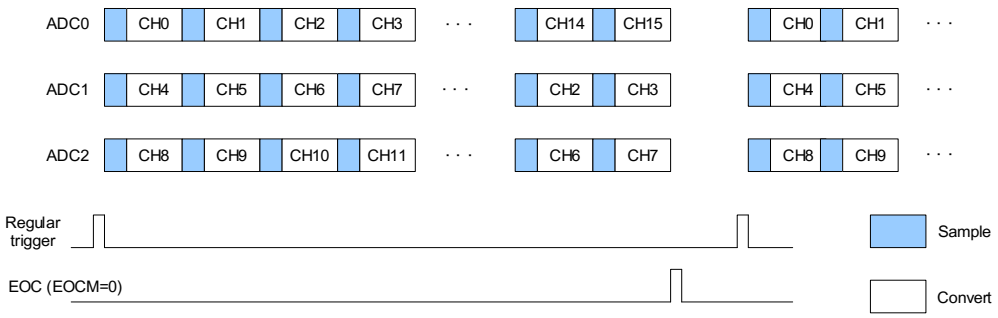
18.5.2. Routine parallel mode

The routine parallel mode is enabled by setting the SYNCM[4:0] bits in the ADC_SYNCCTL register to 5b'00110 or 5b'10110. In the routine parallel mode, all of the ADCs convert the routine sequence parallelly at the selected external trigger of ADC0. The triggers is selected by configuring the ETSRC[3:0] 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 18-14. Routine parallel mode on 16 channels* .

Figure 18-14. 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).
3. ADC2 works freely if SYNCM=00110.

18.5.3. Routine follow-up mode

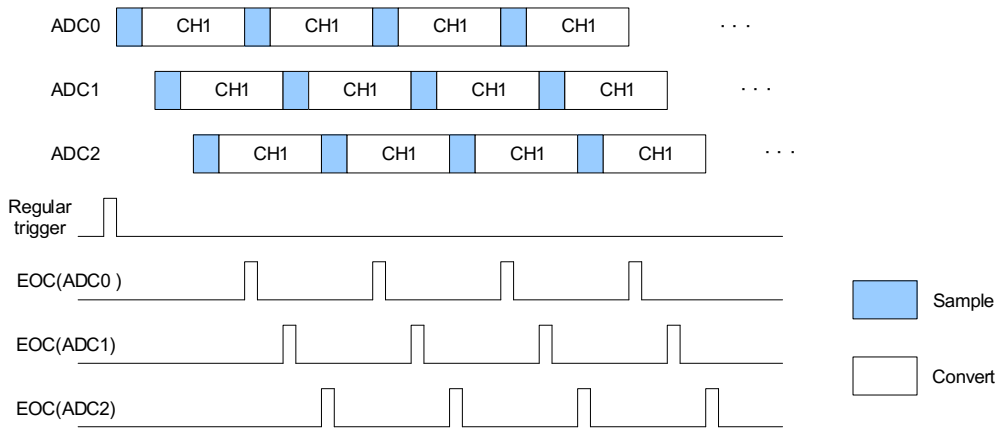
The routine follow-up mode is enabled by setting the SYNCM[4:0] bits in the ADC_SYNCCTL register to 5'b00111 or 5'b10111. In the follow-up mode, ADC0 converts the routine sequence at the selected external trigger. The triggers are selected by configuring the ETSRC[3:0] bits in the ADC_CTL1 register of ADC0. After a delay time, ADC1 converts the routine sequence. After another delay time, ADC2 converts the routine sequence. The routine sequence 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 routine follow-up mode is shown in the *Figure 18-15. Routine follow-up mode on 1 channel*

in continuous operation mode.

Figure 18-15. Routine follow-up mode on 1 channel in continuous operation mode



Note:

1. Make sure to trigger the ADCs when none of them is converting (do not trigger ADC0 when some of the conversions are not finished).
2. ADC2 works freely if SYNCM=00111.

18.5.4. Use DMA in ADC sync mode

In ADC sync mode, the converted data of routine channel are stored in the ADC sync routine data register (ADC_SYNCDATA). DMA can be used to transfer data from ADC_SYNCDATA 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 16. 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_SYNCDATA 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[15:0] -> ADC1_RDATA[15:0] -> ADC0_RDATA[15:0] -> ADC1_RDATA[15:0]. When all of the three ADCs work in SYNC mode, the transfer sequence is ADC0_RDATA[15:0] -> ADC1_RDATA[15:0] -> ADC2_RDATA[15:0] -> ADC0_RDATA[15:0] -> ADC1_RDATA[15:0] -> ADC2_RDATA[15:0].

The ADC Sync DMA mode 0 is properly for:

- ADC0 and ADC1 work in routine parallel mode (SYNCM=5b'00110)
- All ADCs work in routine parallel mode (SYNCM=5b'10110)

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_SYNCDATA 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]}. When all of the three ADCs work in SYNC mode, the transfer sequence is {ADC1_RDATA[15:0],ADC0_RDATA[15:0]} -> {ADC0_RDATA[15:0],ADC2_RDATA[15:0]} ->{ADC2_RDATA[15:0],ADC1_RDATA[15:0]} -> {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=5b'00110)
- ADC0 and ADC1 work in routine follow-up mode (SYNCM=5b'00111)
- All ADCs work in routine follow-up mode (SYNCM=5b'10111)

18.6. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for routine channel or sequence.
- The analog watchdog event.
- Overflow event.

The interrupts of ADC0, ADC1 and ADC2 are mapped into the same interrupt vector ISR[18].

18.7. Register definition

ADC0 base address: 0x4001 2000

ADC1 base address: 0x4001 2100

ADC2 base address: 0x4001 2200

18.7.1. Status register (ADC_STAT)

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										ROVF	STRC	Reserved		EOC	WDE
										rc_w0	rc_w0			rc_w0	rc_w0

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	ROVF	Routine data register overflow 0: Routine data register not overflow 1: Routine data register overflow This bit is set by hardware when the routine 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 data is lost when this bit is set. Cleared by software writing 0 to it.
4	STRC	Start flag of routine sequence conversion 0: Conversion is not started 1: Conversion is started 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 0: No end of routine sequence conversion 1: End ofroutine sequence conversion Set by hardware at the end of a routine sequence conversion. Cleared by software writing 0 to it or by reading the ADC_RDATA register.
0	WDE	Analog watchdog event flag

0: No analog watchdog event

1: Analog watchdog event

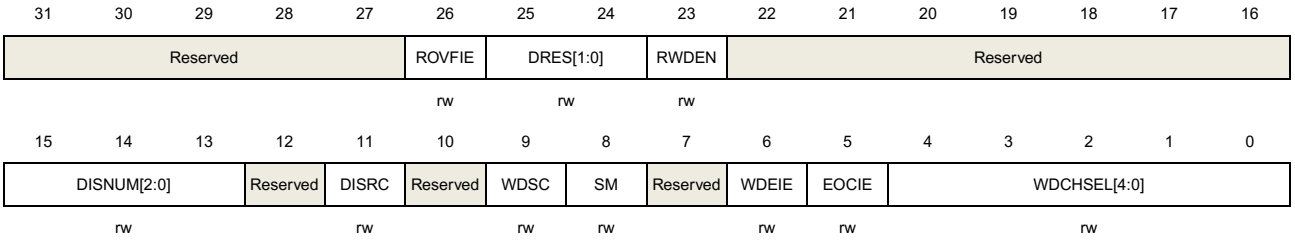
Set by hardware when the converted voltage crosses the values programmed in the ADC_WDLT and ADC_WDHT registers. Cleared by software writing 0 to it.

18.7.2. Control register 0 (ADC_CTL0)

Address offset: 0x04

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	ROVFIE	Interrupt enable for ROVF 0: ROVF interrupt disable 1: ROVF interrupt enable
25:24	DRES[1:0]	ADC data resolution 00: 12bit; 01: 10bit; 10: 8bit; 11: 6bit
23	RWDEN	Routine channel analog watchdog enable 0: Analog watchdog disable 1: Analog watchdog enable
22:16	Reserved	Must be kept at reset value.
15:13	DISNUM[2:0]	Number of conversions in discontinuous mode The number of channels to be converted after a trigger will be DISNUM+1 in routine sequence.
12	Reserved	Must be kept at reset value.
11	DISRC	Discontinuous mode on routine sequence 0: Discontinuous operation mode disable 1: Discontinuous operation mode enable

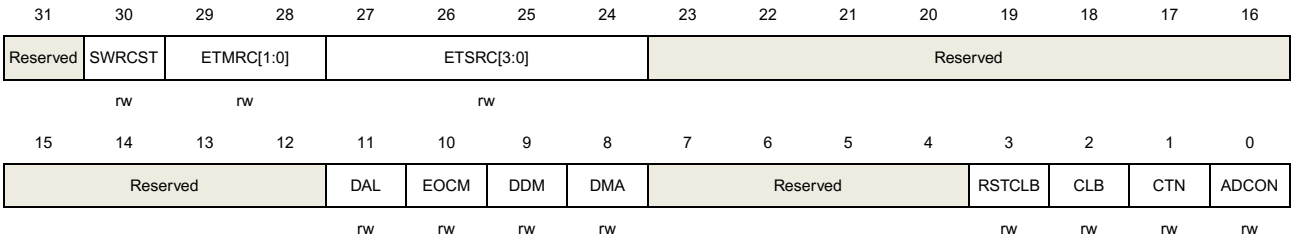
10	Reserved	Must be kept at reset value.
9	WDSC	When in scan mode, analog watchdog is effective on a single channel 0: All channels have analog watchdog function 1: A single channel has analog watchdog function
8	SM	Scan mode 0: Scan operation mode disable 1: Scan operation mode enable
7	Reserved	Must be kept at reset value.
6	WDEIE	Interrupt enable for WDE 0: Interrupt disable 1: Interrupt enable
5	EOCIE	Interrupt enable for EOC 0: Interrupt disable 1: Interrupt enable
4:0	WDCHSEL[4:0]	Analog watchdog channel select 00000: ADC channel0 00001: ADC channel1 00010: ADC channel2 00011: ADC channel 3 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 Other values are reserved. **Note:** ADC0 analog inputs Channel16, Channel17 and Channel 18 are internally connected to the temperature sensor, to V_{REFINT} and to V_{BAT} analog inputs. ADC1 analog inputs Channel16, Channel17 and Channel18 are internally connected to V_{SSA}. ADC2 analog inputs Channel16, Channel17 and Channel18 are internally connected to V_{SSA}.

18.7.3. Control register 1 (ADC_CTL1)

Address offset: 0x08

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	SWRCST	Software start on routine sequence. Setting 1 on this bit starts a conversion of a routine sequence. 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 00: External trigger for routine sequence disable 01: Rising edge of external trigger for routine sequence enable 01: Falling edge of external trigger for routine sequence enable 11: Rising and falling edge of external trigger for routine sequence enable
27:24	ETSRC[3:0]	External trigger select for routine sequence 0000: Timer 0 CH0 0001: Timer 0 CH1 0010: Timer 0 CH2 0011: Timer 1 CH1 0100: Timer 1 CH2 0101: Timer 1 CH3 0110: Timer 1 TRGO 0111: Timer 2 CH0 1000: Timer 2 TRGO 1001: Timer 3 CH3 1010: Timer 4 CH0 1011: Timer 4 CH1 1100: Timer 4 CH2 1101: Timer 7 CH0 1110: Timer 7 TRGO 1111: EXTI line 11
23:12	Reserved	Must be kept at reset value.
11	DAL	Data alignment

		0: LSB alignment 1: MSB alignment
10	EOCM	End of conversion mode 0: Only at the end of a sequence of routine conversions, the EOC bit is set. Overflow detection is disabled unless DMA=1. 1: At the end of each routine conversion, the EOC bit is set. Overflow is detected automatically
9	DDM	DMA disable mode This bit configure the DMA disable mode for single ADC mode 0: The DMA engine is disabled after the end of transfer signal from DMA controller is detected. 1: When DMA=1, the DMA engine issues a request at end of each routine conversion.
8	DMA	DMA request enable. 0: DMA request disable 1: DMA request enable
7:4	Reserved	Must be kept at reset value
3	RSTCLB	Reset calibration This bit is set by software and cleared by hardware after the calibration registers are initialized. 0: Calibration register initialize done. 1: Initialize calibration register start
2	CLB	ADC calibration 0: Calibration done 1: Calibration start
1	CTN	Continuous mode 0: Continuous operation mode disable 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. 0: ADC disable and power down 1: ADC enable

18.7.4. Sample time register 0 (ADC_SAMPT0)

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	SPT18[2:0]	SPT17[2:0]	SPT16[2:0]	SPT15[2:1]
	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPT15[0]	SPT14[2:0]			SPT13[2:0]			SPT12[2:0]			SPT11[2:0]			SPT10[2:0]		
rw	rw			rw			rw			rw			rw		

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value
26:24	SPT18[2:0]	refer to SPT10[2:0] description
23:21	SPT17[2:0]	refer to SPT10[2:0] description
20:18	SPT16[2:0]	refer to SPT10[2:0] description
17:15	SPT15[2:0]	refer to SPT10[2:0] description
14:12	SPT14[2:0]	refer to SPT10[2:0] description
11:9	SPT13[2:0]	refer to SPT10[2:0] description
8:6	SPT12[2:0]	refer to SPT10[2:0] description
5:3	SPT11[2:0]	refer to SPT10[2:0] description
2:0	SPT10[2:0]	Channel sample time 000: channel sampling time is 3 cycles 001: channel sampling time is 15 cycles 010: channel sampling time is 28 cycles 011: channel sampling time is 56 cycles 100: channel sampling time is 84 cycles 101: channel sampling time is 112 cycles 110: channel sampling time is 144 cycles 111: channel sampling time is 480 cycles

18.7.5. Sample time register 1 (ADC_SAMPT1)

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		SPT9[2:0]			SPT8[2:0]			SPT7[2:0]			SPT6[2:0]			SPT5[2:1]	
		rw			rw			rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPT5[0]	SPT4[2:0]			SPT3[2:0]			SPT2[2:0]			SPT1[2:0]			SPT0[2:0]		
rw	rw			rw			rw			rw			rw		

Bits	Fields	Descriptions

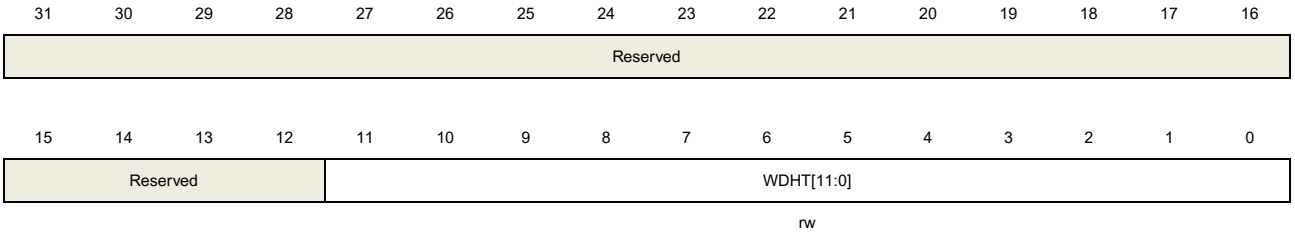
31:30	Reserved	Must be kept at reset value
29:27	SPT9[2:0]	refer to SPT0[2:0] description
26:24	SPT8[2:0]	refer to SPT0[2:0] description
23:21	SPT7[2:0]	refer to SPT0[2:0] description
20:18	SPT6[2:0]	refer to SPT0[2:0] description
17:15	SPT5[2:0]	refer to SPT0[2:0] description
14:12	SPT4[2:0]	refer to SPT0[2:0] description
11:9	SPT3[2:0]	refer to SPT0[2:0] description
8:6	SPT2[2:0]	refer to SPT0[2:0] description
5:3	SPT1[2:0]	refer to SPT0[2:0] description
2:0	SPT0[2:0]	Channel sample time 000: channel sampling time is 3 cycles 001: channel sampling time is 15 cycles 010: channel sampling time is 28 cycles 011: channel sampling time is 56 cycles 100: channel sampling time is 84 cycles 101: channel sampling time is 112 cycles 110: channel sampling time is 144 cycles 111: channel sampling time is 480 cycles

18.7.6. Watchdog high threshold register (ADC_WDHT)

Address offset: 0x24

Reset value: 0x0000 0FFF

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



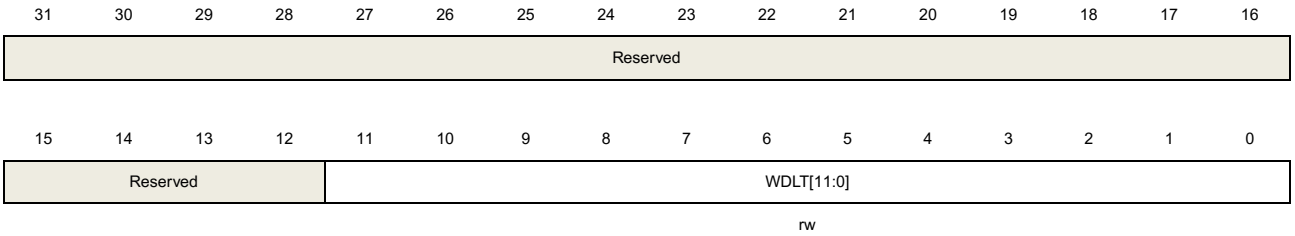
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	WDHT[11:0]	High threshold for analog watchdog These bits define the high threshold for the analog watchdog.

18.7.7. Watchdog low threshold register (ADC_WDLT)

Address offset: 0x28

Reset value: 0x0000 0000

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



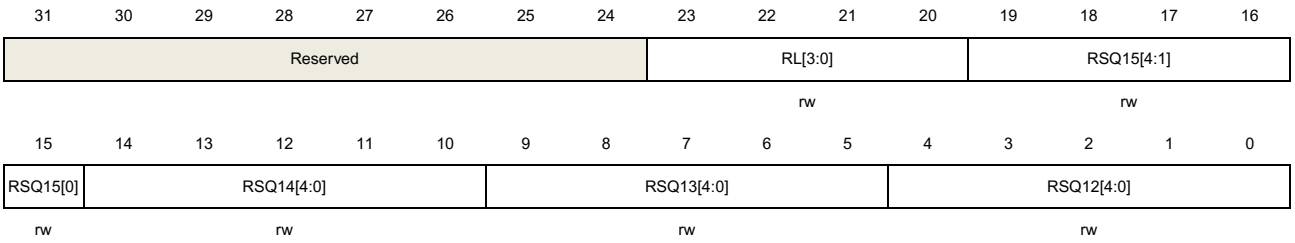
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	WDLT[11:0]	Low threshold for analog watchdog These bits define the low threshold for the analog watchdog.

18.7.8. Routine sequence register 0 (ADC_RSQ0)

Address offset: 0x2C

Reset value: 0x0000 0000

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



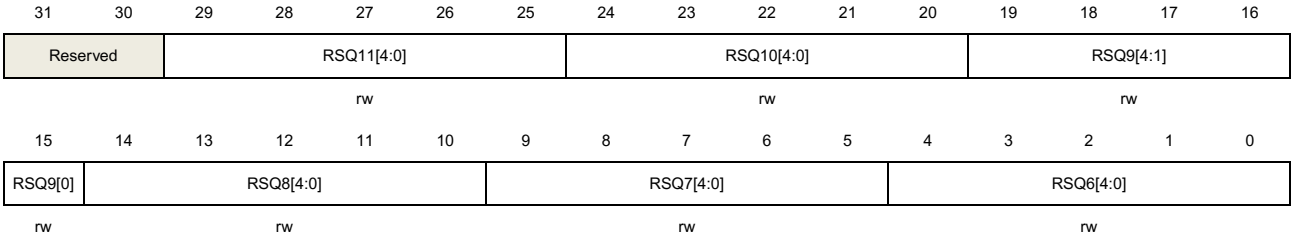
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:20	RL[3:0]	Routine sequence length. The total number of conversion in routine sequence equals to RL[3:0]+1.
19:15	RSQ15[4:0]	refer to RSQ0[4:0] description
14:10	RSQ14[4:0]	refer to RSQ0[4:0] description
9:5	RSQ13[4:0]	refer to RSQ0[4:0] description
4:0	RSQ12[4:0]	refer to RSQ0[4:0] description

18.7.9. Routine sequence register 1 (ADC_RSQ1)

Address offset: 0x30

Reset value: 0x0000 0000

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



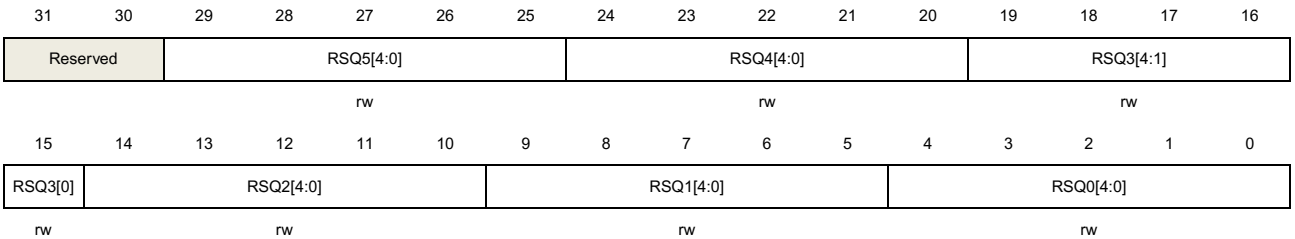
Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:25	RSQ11[4:0]	refer to RSQ0[4:0] description
24:20	RSQ10[4:0]	refer to RSQ0[4:0] description
19:15	RSQ9[4:0]	refer to RSQ0[4:0] description
14:10	RSQ8[4:0]	refer to RSQ0[4:0] description
9:5	RSQ7[4:0]	refer to RSQ0[4:0] description
4:0	RSQ6[4:0]	refer to RSQ0[4:0] description

18.7.10. Routine sequence register 2 (ADC_RSQ2)

Address offset: 0x34

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:25	RSQ5[4:0]	refer to RSQ0[4:0] description
24:20	RSQ4[4:0]	refer to RSQ0[4:0] description
19:15	RSQ3[4:0]	refer to RSQ0[4:0] description

14:10	RSQ2[4:0]	refer to RSQ0[4:0] description
9:5	RSQ1[4:0]	refer to RSQ0[4:0] description
4:0	RSQ0[4:0]	The channel number (0..18) is written to these bits to select a channel as the nth conversion in the routine sequence.

18.7.11. Routine data register (ADC_RDATA)

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
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]	Routine channel data These bits contain routine channel conversion value, which is read only.

18.7.12. Oversample control register (ADC_OVSAMPCTL)

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
Reserved						TOVS	OVSS[3:0]				OVSR[2:0]			Reserved	OVSEN
						rw	rw				rw				rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	TOVS	Triggered Oversampling This bit is set and cleared by software. 0: All oversampled conversions for a channel are done consecutively after a trigger 1: Each conversion needs a trigger for a oversampled channel and the number of triggers is determined by the oversampling ratio(OVSR[2:0]).

		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 This bit is set and cleared by software. 0000: No shift 0001: Shift 1-bit 0010: Shift 2-bits 0011: Shift 3-bits 0100: Shift 4-bits 0101: Shift 5-bits 0110: Shift 6-bits 0111: Shift 7-bits 1000: Shift 8-bits Other codes reserved **Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress).
4:2	OVSR[2:0]	Oversampling ratio This bit filed defines the number of oversampling ratio. 000: 2x 001: 4x 010: 8x 011: 16x 100: 32x 101: 64x 110: 128x 111: 256x **Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress).
1	Reserved	Must be kept at reset value.
0	OVSEN	Oversampler Enable This bit is set and cleared by software. 0: Oversampler disabled 1: Oversampler enabled **Note:** The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress).

18.7.13. Summary status register (ADC_SSTAT)

Address offset: 0x300(for ADC0 base address)
Reset value: 0x0000 0000

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

This register is read only and provides a summary of the three ADCs. This register is not available in ADC1 and ADC2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										ROVF2	STRC2	Reserved		EOC2	WDE2
										r	r			r	r

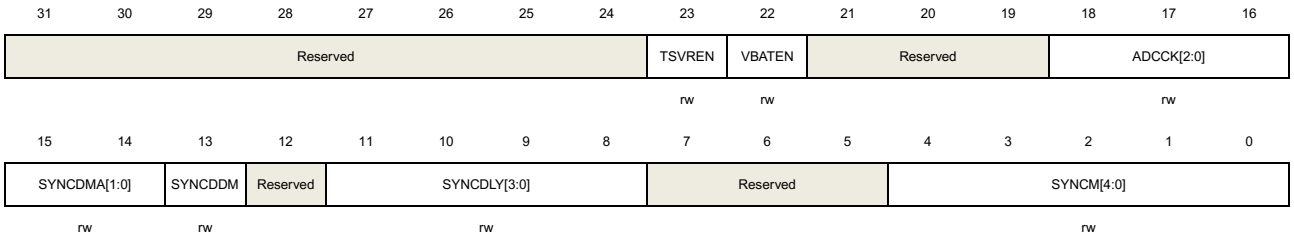
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ROVF1	STRC1	Reserved		EOC1	WDE1	Reserved		ROVF0	STRC0	Reserved		EOC0	WDE0
		r	r			r	r			r	r			r	r

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21	ROVF2	This bit is the mirror image of the ROVF bit of ADC2
20	STRC2	This bit is the mirror image of the STRC bit of ADC2
19:18	Reserved	Must be kept at reset value.
17	EOC2	This bit is the mirror image of the EOC bit of ADC2
16	WDE2	This bit is the mirror image of the WDE bit of ADC2
15:14	Reserved	Must be kept at reset value.
13	ROVF1	This bit is the mirror image of the ROVF bit of ADC1
12	STRC1	This bit is the mirror image of the STRC bit of ADC1
11:10	Reserved	Must be kept at reset value.
9	EOC1	This bit is the mirror image of the EOC bit of ADC1
8	WDE1	This bit is the mirror image of the WDE bit of ADC1
7:6	Reserved	Must be kept at reset value.
5	ROVF0	This bit is the mirror image of the ROVF bit of ADC0
4	STRC0	This bit is the mirror image of the STRC bit of ADC0
3:2	Reserved	Must be kept at reset value.
1	EOC0	This bit is the mirror image of the EOC bit of ADC0
0	WDE0	This bit is the mirror image of the WDE bit of ADC0

18.7.14. Sync control register (ADC_SYNCCTL)

Address offset: 0x304(for ADC0 base address)
Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).
This register is not available in ADC1 and ADC2.



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	TSVREN	Channel 16 (temperature sensor) and 17 (internal reference voltage) enable of ADC0. 0: Channel 16 and 17 of ADC0 disable 1: Channel 16 and 17 of ADC0 enable
22	VBATEN	Channel 18 (1/4 voltage of external battery) enable of ADC0. 0: Channel 18 of ADC0 disable 1: Channel 18 of ADC0 enable
21:19	Reserved	Must be kept at reset value.
18:16	ADCCK[2:0]	ADC clock. These bits configure the ADC clock for all the ADCs. 3'b000:PCLK2 div2; 3'b001:PCLK2 div4; 3'b010:PCLK2 div6; 3'b011:PCLK2 div8; 3'b100:HCLK div5; 3'b101:HCLK div6; 3'b110:HCLK div10; 3'b111:HCLK div20.
15:14	SYNCDMA[1:0]	ADC sync DMA mode selection 00: ADC sync DMA disabled 01: ADC sync DMA mode 0 10: ADC sync DMA mode 1 11: reserved
13	SYNCDDM	ADC sync DMA disable mode This bit configures the DMA disable mode for ADC sync mode 0: The DMA engine is disabled after the end of transfer signal from DMA controller is detected. 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

		These bits are used to configure the delay between 2 sampling phases in ADC sync modes to (5+SYNCDLY) ADC clock cycles.
7:5	Reserved	Must be kept at reset value.
4:0	SYNCM[4:0]	ADC sync mode When ADC sync mode is enabled these bits should be set to 00000 firstly before change to another value. 5'b00000: ADC sync mode disabled. All the ADCs work independently. 5'b00110: ADC0 and ADC1 work in routine parallel mode. ADC2 works independently. 5'b00111: ADC0 and ADC1 work in routine follow-up mode. ADC2 works independently. 5'b10110: All ADCs work in routine parallel mode 5'b10111: All ADCs work in routine follow-up mode All other values are reserved.

18.7.15. Sync routine data register (ADC_SYNCDATA)

Address offset: 0x308(for ADC0 base address)
Reset value: 0x0000 0000

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

This register is not available in ADC1 and ADC2.

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 data2 in ADC sync mode
15:0	SYNCDATA0[15:0]	Routine data1 in ADC sync mode

19. Digital-to-analog converter (DAC)

19.1. 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.

The DAC channels can work independently or concurrently.

19.2. 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}.
- Noise wave generation (LFSR noise mode and triangle noise mode).
- Two DAC channels in concurrent mode.

Figure 19-1. DAC block diagram and *Table 19-1. DAC I/O description* show the block diagram of DAC and the pin description of DAC, respectively.

Figure 19-1. DAC block diagram

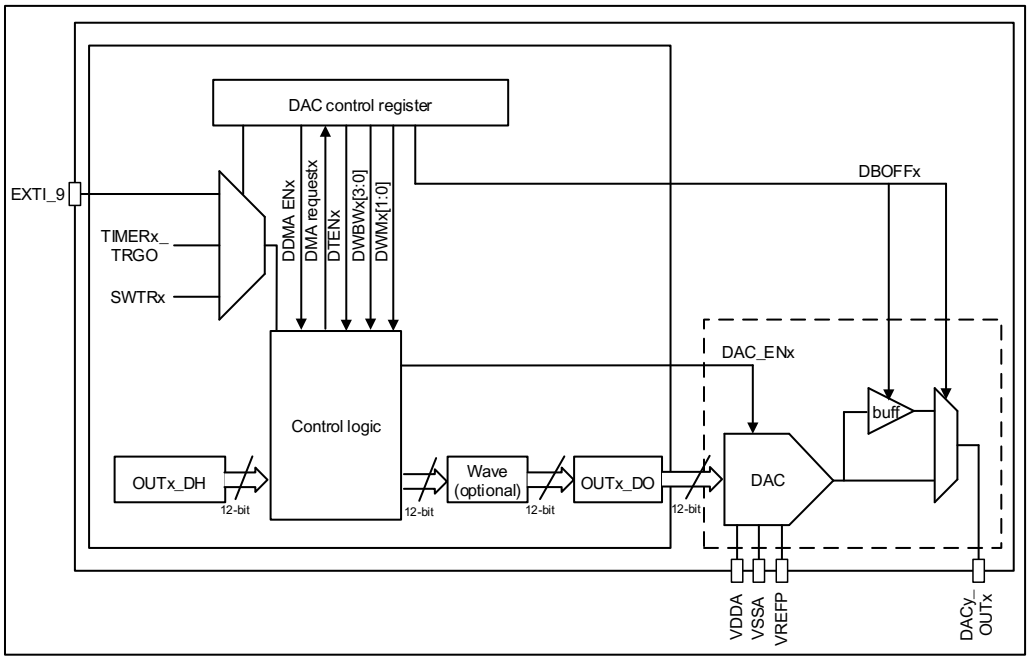


Table 19-1. DAC I/O description

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 19-2. DAC triggers and outputs summary

	DAC0	
Channel	**Channel0**	**Channel1**
DAC outputs connected to I / Os	PA4	PA5
DAC output buffer	●	●
DAC software trigger	●	
DAC trigger signals from EXTI	EXTI_9	
DAC trigger signals from TIMER	TIMER1_TRGO TIMER3_TRGO TIMER4_TRGO TIMER5_TRGO TIMER6_TRGO TIMER7_TRGO	

Note: The GPIO pins should be configured to analog mode before enable the DAC module.

19.3. Function overview

19.3.1. 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.

19.3.2. 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.

19.3.3. DAC data configuration

The 12-bit DAC holding data (OUTx_DH) can be configured by writing any one of the DAC_OUTx_R12DH, DAC_OUTx_L12DH and DAC_OUTx_R8DH registers. When the data is loaded by DAC_OUTx_R8DH register, only the MSB 8 bits are configurable, the LSB 4 bits are forced to 4'b0000.

19.3.4. 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 19-3. Triggers of DAC***.

Table 19-3. Triggers of DAC

DTSELx[2:0]	Trigger Source	Trigger Type
3b'000	TIMER5_TRGO	Hardware trigger
3b'001	TIMER7_TRGO	
3b'010	TIMER6_TRGO	
3b'011	TIMER4_TRGO	
3b'100	TIMER1_TRGO	
3b'101	TIMER3_TRGO	
3b'110	EXTI_9	
3b'111	SWTR	Software trigger

The TIMERx_TRGO signals are generated from the timers, while the software trigger can be generated by setting the SWTRx bits in the DAC_SWT register.

19.3.5. 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 (DAC_OUTx_DO) register when the selected trigger event happened. When the external trigger is disabled, the transfer is performed automatically.

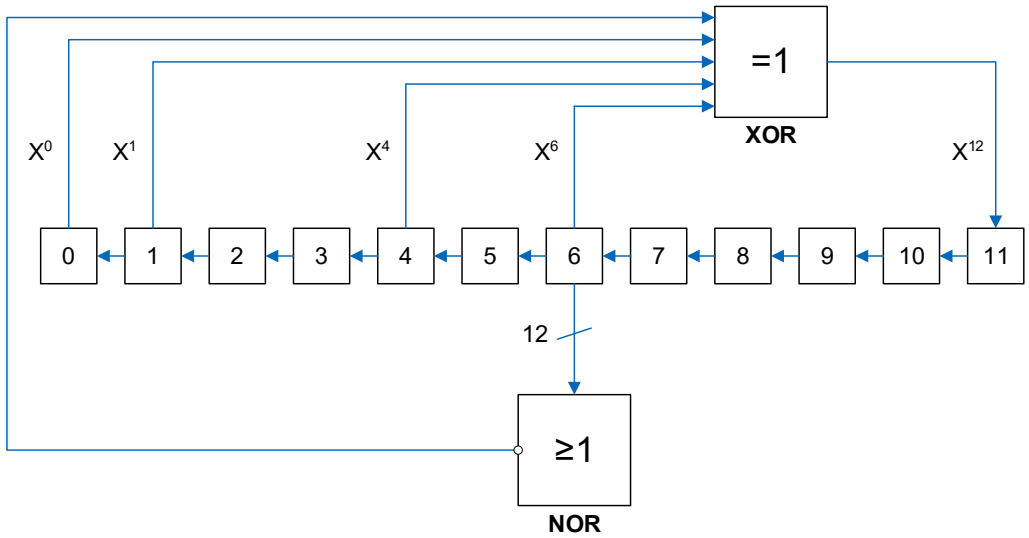
When the DAC holding data (OUTx_DH) is loaded into the DAC_OUTx_DO register, after the time $t_{SETTLING}$ which is determined by the analog output load and the power supply voltage, the analog output is valid.

19.3.6. 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.

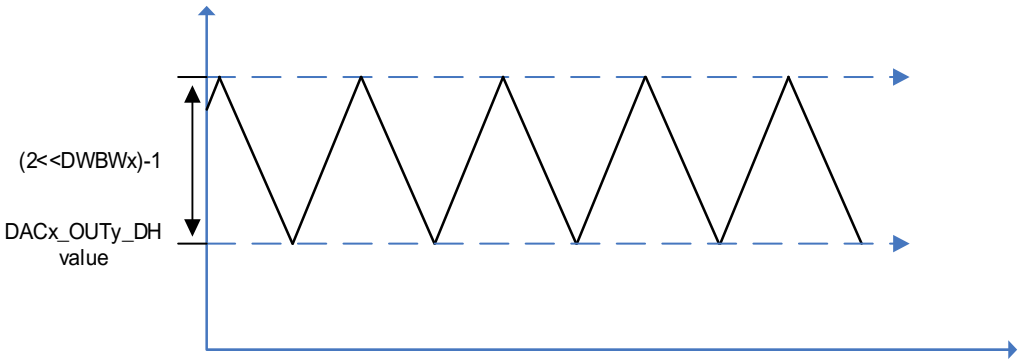
LFSR noise wave mode: there is a Linear Feedback Shift Register (LFSR) in the DAC control logic, it controls the LFSR noise signal which is added to the OUTx_DH value, and then the result is stored into the DAC_OUTx_DO register When the configured DAC noise wave bit width is less than 12, the noise signal equals to the LSB DWBWx bits of the LFSR register, while the MSB bits are masked.

Figure 19-2. DAC LFSR algorithm



Triangle noise mode: a triangle signal is added to the OUTx_DH value, and then the result is stored into the DAC_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 19-3. DAC triangle noise wave



19.3.7. DAC output voltage

The following equation determines the analog output voltage on the DAC pin.

$$V_{DAC_OUT}=V_{REFP}*OUTx_DO/4096 \quad (19\text{-}1)$$

The digital input is linearly converted to an analog output voltage and its range is 0 to V_{REFP}.

19.3.8. 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.

19.3.9. 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.

19.4. Register definition

DAC0 base address: 0x4000 7400

19.4.1. 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		DDUDRIE1	DDMAEN1	DWBW1[3:0]				DWM1[1:0]		DTSEL1[2:0]			DTEN1	DBOFF1	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		DDUDRIE0	DDMAEN0	DWBW0[3:0]				DWM0[1:0]		DTSEL0[2:0]			DTEN0	DBOFF0	DEN0
		rw	rw	rw				rw		rw			rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DDUDRIE1	DACx_OUT1 DMA underrun interrupt enable 0: DACx_OUT1 DMA underrun interrupt disabled 1: DACx_OUT1 DMA underrun interrupt enabled
28	DDMAEN1	DACx_OUT1 DMA enable 0: DACx_OUT1 DMA mode disabled 1: DACx_OUT1 DMA mode enabled
27:24	DWBW1[3:0]	DACx_OUT1 noise wave bit width 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. 0000: The bit width of the wave signal is 1 0001: The bit width of the wave signal is 2 0010: The bit width of the wave signal is 3 0011: The bit width of the wave signal is 4 0100: The bit width of the wave signal is 5 0101: The bit width of the wave signal is 6 0110: The bit width of the wave signal is 7 0111: The bit width of the wave signal is 8 1000: The bit width of the wave signal is 9 1001: The bit width of the wave signal is 10 1010: The bit width of the wave signal is 11

		>=1011: The bit width of the wave signal is 12
23:22	DWM1[1:0]	DACx_OUT1 noise wave mode These bits specify the mode selection of the noise wave signal of DACx_OUT1 when external trigger of DACx_OUT1 is enabled (DTEN1=1). 00: wave disabled 01: LFSR noise mode 1x: Triangle noise mode
21:19	DTSEL1[2:0]	DACx_OUT1 trigger selection These bits are only used if bit DTEN = 1 and select the external event used to trigger DAC. 000: TIMER5 TRGO 001: TIMER7 TRGO 010: TIMER6 TRGO 011: TIMER4 TRGO 100: TIMER1 TRGO 101: TIMER3 TRGO 110: EXTI line 9 111: Software trigger
18	DTEN1	DACx_OUT1 trigger enable 0: DACx_OUT1 trigger disabled 1: DACx_OUT1 trigger enabled
17	DBOFF1	DACx_OUT1 output buffer turn off 0: DACx_OUT1 output buffer turns on to reduce the output impedance and improve the driving capability 1: DACx_OUT1 output buffer turns off
16	DEN1	DACx_OUT1 enable 0: DACx_OUT1 disabled 1: DACx_OUT1 enabled
15:14	Reserved	Must be kept at reset value
13	DDUDRIE0	DACx_OUT0 DMA underrun interrupt enable 0: DACx_OUT0 DMA underrun interrupt disabled 1: DACx_OUT0 DMA underrun interrupt enabled
12	DDMAEN0	DACx_OUT0 DMA enable 0: DACx_OUT0 DMA mode disabled 1: DACx_OUT0 DMA mode enabled
11:8	DWBW0[3:0]	DACx_OUT0 noise wave bit width 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

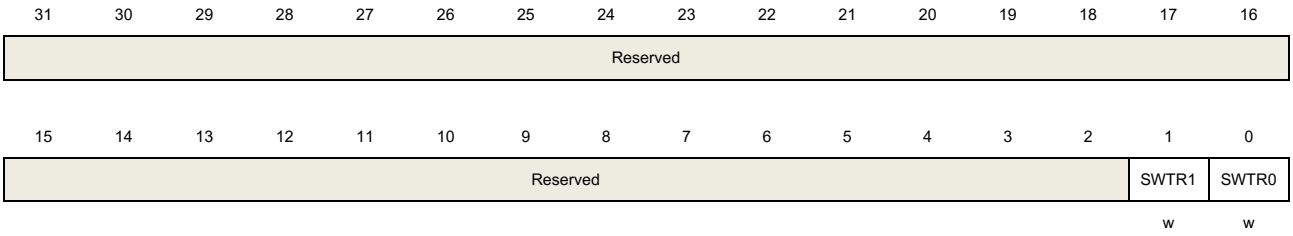
		0001: The bit width of the wave signal is 2 0010: The bit width of the wave signal is 3 0011: The bit width of the wave signal is 4 0100: The bit width of the wave signal is 5 0101: The bit width of the wave signal is 6 0110: The bit width of the wave signal is 7 0111: The bit width of the wave signal is 8 1000: The bit width of the wave signal is 9 1001: The bit width of the wave signal is 10 1010: The bit width of the wave signal is 11 >=1011: The bit width of the wave signal is 12
7:6	DWM0[1:0]	DACx_OUT0 noise wave mode These bits specify the mode selection of the noise wave signal of DACx_OUT0 when external trigger of DACx_OUT0 is enabled (DTEN0=1). 00: Wave disabled 01: LFSR noise mode 1x: Triangle noise mode
5:3	DTSEL0[2:0]	DACx_OUT0 trigger selection These bits are only used if bit DTEN = 1 and select the external event used to trigger DAC. 000: TIMER5 TRGO 001: TIMER7 TRGO 010: TIMER6 TRGO 011: TIMER4 TRGO 100: TIMER1 TRGO 101: TIMER3 TRGO 110: EXTI line 9 111: Software trigger
2	DTEN0	DACx_OUT0 trigger enable 0: DACx_OUT0 trigger disabled 1: DACx_OUT0 trigger enabled
1	DBOFF0	DACx_OUT0 output buffer turn off 0: DACx_OUT0 output buffer turns on to reduce the output impedance and improve the driving capability 1: DACx_OUT0 output buffer turns off
0	DEN0	DACx_OUT0 enable 0: DACx_OUT0 disabled 1: DACx_OUT0 enabled

19.4.2. DACx software trigger register (DAC_SWT)

Address offset: 0x04

Reset value: 0x0000 0000

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



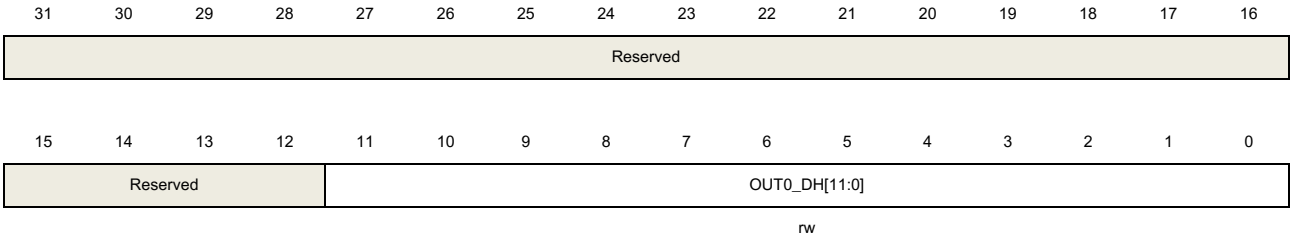
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	SWTR1	DACx_OUT1 software trigger, cleared by hardware. 0: Software trigger disabled 1: Software trigger enabled
0	SWTR0	DACx_OUT0 software trigger, cleared by hardware. 0: Software trigger disabled 1: Software trigger enabled

19.4.3. 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).



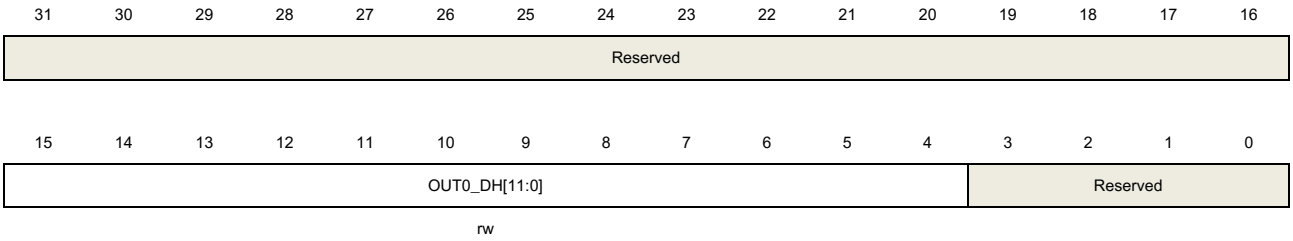
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT0_DH[11:0]	DACx_OUT0 12-bit right-aligned data. These bits specify the data that is to be converted by DACx_OUT0.

19.4.4. 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).



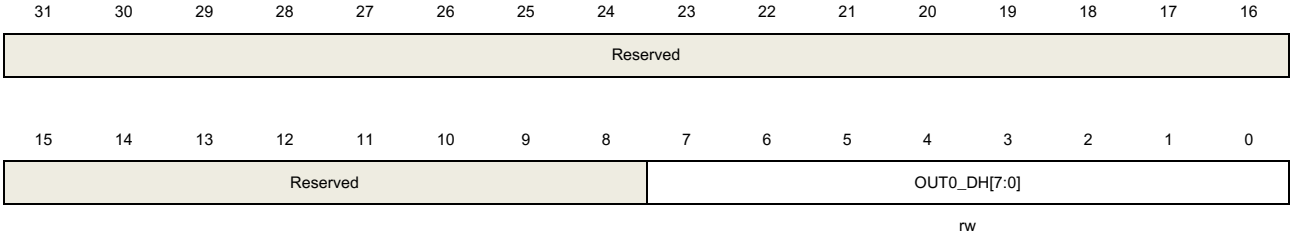
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	OUT0_DH[11:0]	DACx_OUT0 12-bit left-aligned data. These bits specify the data that is to be converted by DACx_OUT0.
3:0	Reserved	Must be kept at reset value.

19.4.5. 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).



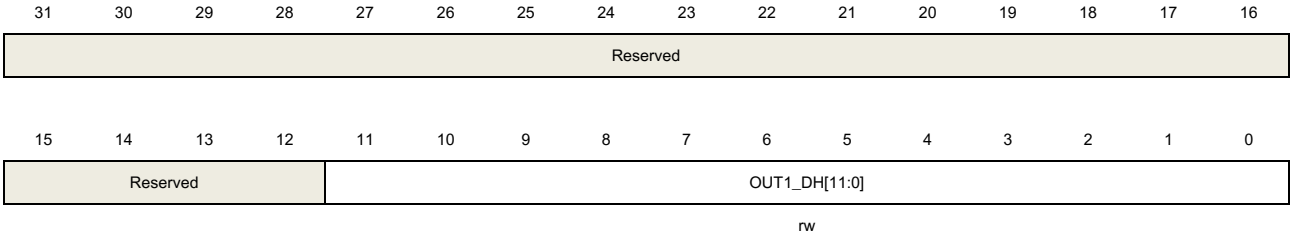
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	OUT0_DH[7:0]	DACx_OUT0 8-bit right-aligned data. These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT0.

19.4.6. 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).



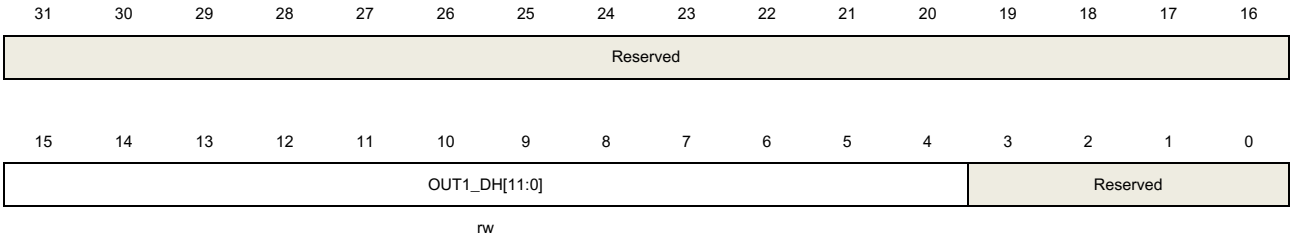
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT1_DH[11:0]	DACx_OUT1 12-bit right-aligned data. These bits specify the data that is to be converted by DACx_OUT1.

19.4.7. 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).



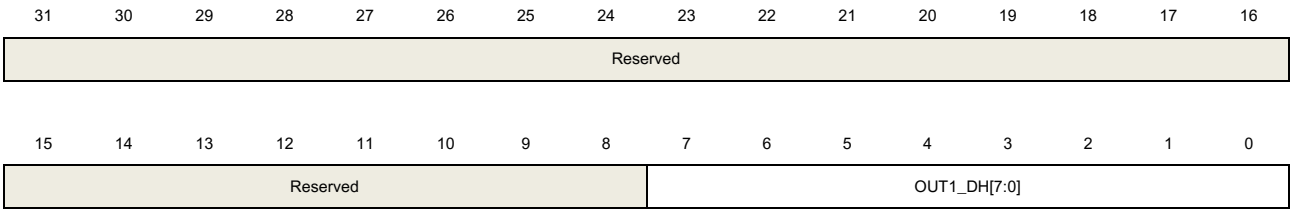
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	OUT1_DH[11:0]	DACx_OUT1 12-bit left-aligned data. These bits specify the data that is to be converted by DACx_OUT1.
3:0	Reserved	Must be kept at reset value.

19.4.8. 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).



rw

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.

19.4.9. 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).



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 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 These bits specify the data that is to be converted by DACx_OUT0.

19.4.10. 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)



rw

Bits	Fields	Descriptions
31:20	OUT1_DH[11:0]	DACx_OUT1 12-bit left-aligned data 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 These bits specify the data that is to be converted by DACx_OUT0.
3:0	Reserved	Must be kept at reset value.

19.4.11. 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).

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
OUT1_DH [7:0]								OUT0_DH [7:0]							
rw								rw							

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 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 These bits specify the MSB 8-bit of the data that is to be converted by DACx_OUT0.

19.4.12. 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).

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_DO [11:0]
	r

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT0_DO [11:0]	DACx_OUT0 12-bit output data These bits, which are read only, storage the data that is being converted by DACx_OUT0.

19.4.13. 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).

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				OUT1_DO [11:0]											
				r											

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	OUT1_DO [11:0]	DACx_OUT1 12-bit output data These bits, which are read only, storage the data that is being converted by DACx_OUT1.

19.4.14. DACx status register 0 (DAC_STAT0)

Address offset: 0x34

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DDUDR1	Reserved												
		rc_w1													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DDUDR0	Reserved												
		rc_w1													

Bits	Fields	Descriptions

31:30	Reserved	Must be kept at reset value.
29	DDUDR1	DACx_OUT1 DMA underrun flag. This bit is set by hardware and cleared by software (by writing it to 1). 0: No underrun occurred. 1: Underrun occurred (Speed of DAC trigger is high than the DMA transfer).
28:14	Reserved	Must be kept at reset value.
13	DDUDR0	DACx_OUT0 DMA underrun flag. This bit is set by hardware and cleared by software (by writing it to 1). 0: No underrun occurred. 1: Underrun occurred (Speed of DAC trigger is high than the DMA transfer).
12:0	Reserved	Must be kept at reset value.

20. 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 when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

20.1. Free watchdog timer (FWDGT)

20.1.1. 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 causes a reset when the internal down counter reaches 0. The register write protection function in free watchdog timer can be enabled to prevent it from changing the configuration unexpectedly.

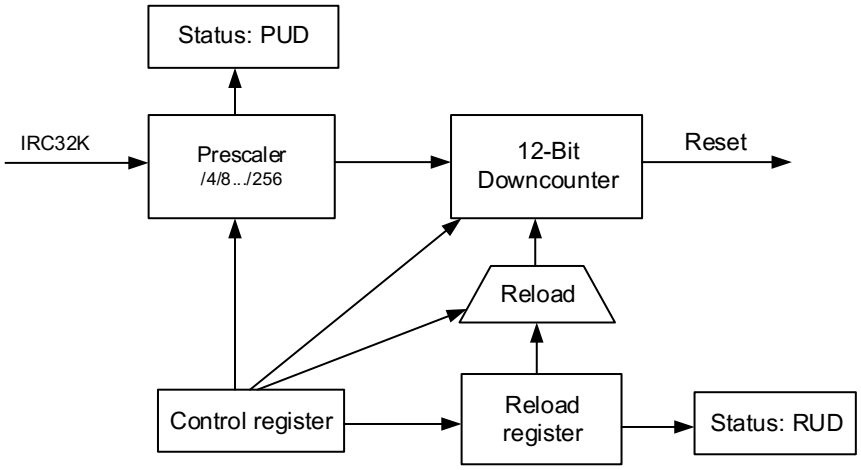
20.1.2. Characteristics

- Free-running 12-bit downcounter.
- Reset when the downcounter reaches 0, if the watchdog is enabled.
- Free clock source，FWDGT can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog timer bit, automatically start the FWDGT at power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.

20.1.3. Function overview

The free watchdog timer consists of an 8-stage prescaler and a 12-bit down-counter. ***Figure 20-1. Free watchdog timer block diagram*** shows the functional block of the free watchdog timer module.

Figure 20-1. Free watchdog timer block diagram



The free watchdog timer is enabled by writing the value 0xCCCC to the control register (FWDGT_CTL), then the 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 anytime. 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.

The free watchdog timer can automatically start when power on if the hardware free watchdog timer 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 and the FWDGT_RLD 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) 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®-M33 core halted (Debug mode). The FWDGT stops in Debug mode if the FWDGT_HOLD bit is set.

Table 20-1. Min/max FWDGT timeout period at 32 kHz (IRC32K)

Prescaler divider	PSC[3:0] bits	Min timeout (ms) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFF
1 / 4	0000	0.125	511.9
1 / 8	0001	0.25	1023.8
1 / 16	0010	0.5	2047.5
1 / 32	0011	1.0	4095.0
1 / 64	0100	2.0	8190.0
1 / 128	0101	4.0	16380.0

Prescaler divider	PSC[3:0] bits	Min timeout (ms) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFF
1 / 256	0110	8.0	32760.0
1 / 512	0111	16	65520.0
1 / 1024	1000	32	131040.0
1 / 2048	1001	64	262080.0
1 / 4096	1010	128	524160.0
1 / 8192	1011	256	1048320.0
1 / 16384	1100	512	2096640.0
1 / 32768	1101~1111	1024	4193280.0

The FWDGT timeout can be more accurate by calibrating the IRC32K.

Note: 1. 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. 2. There must be an interval of 7 or more IRC32K clock cycles between two consecutive watchdog reload operations.

20.1.4. Register definition

FWDGT base address: 0x4000 3000

Control register (FWDGT_CTL)

Address offset: 0x00
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
CMD[15:0]															
w															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CMD[15:0]	Write only. Several different fuctions are realized by writing these bits with different values: 0x5555: Disable the FWDGT_PSC and FWDGT_RLD write protection 0xCCCC: Start the free watchdog timer counter. When the counter reduces to 0, the free watchdog timer generates a reset 0xAAAA: Reload the counter.

Prescaler register (FWDGT_PSC)

Address offset: 0x04
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												PSC[3:0]			
												rw			

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
3:0	PSC[3:0]	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. 0000: 1 / 4 0001: 1 / 8 0010: 1 / 16

0011: 1 / 32
0100: 1 / 64
……
1100: 1 / 16384
1101~1111: 1 / 32768
If several prescaler values are used by the application, it is mandatory to wait until PUD bit is reset before changing the prescaler value. However, after updating the prescaler value it is not necessary to wait until PUD is reset before continuing code execution except in case of low-power mode entry.

Reload register (FWDGT_RLD)

Address offset: 0x08
Reset value: 0x0000 0FFF

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				RLD [11:0]											
				rw											

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 counter with the RLD value. These bits are write-protected. Write 0x5555 in 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. If several reload values are used by the application, it is mandatory to wait until RUD bit is reset before changing the reload value. However, after updating the reload value it is not necessary to wait until RUD is reset before continuing code execution except in case of low-power mode entry.

Status register (FWDGT_STAT)

Address offset: 0x0C
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														RUD	PUD
														r	r

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	RUD	Free watchdog timer counter reload value update During a write operation to FWDGT_RLD register, this bit is set and the value read from FWDGT_RLD register is invalid. This bit is reset by hardware after the update operation of FWDGT_RLD register.
0	PUD	Free watchdog timer prescaler value update During a write operation to FWDGT_PSC register, this bit is set and the value read from FWDGT_PSC register is invalid. This bit is reset by hardware after the update operation of FWDGT_PSC register.

20.2. Window watchdog timer (WWDGT)

20.2.1. 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 APB1 clock. The window watchdog timer is suitable for the situation that requires an accurate timing.

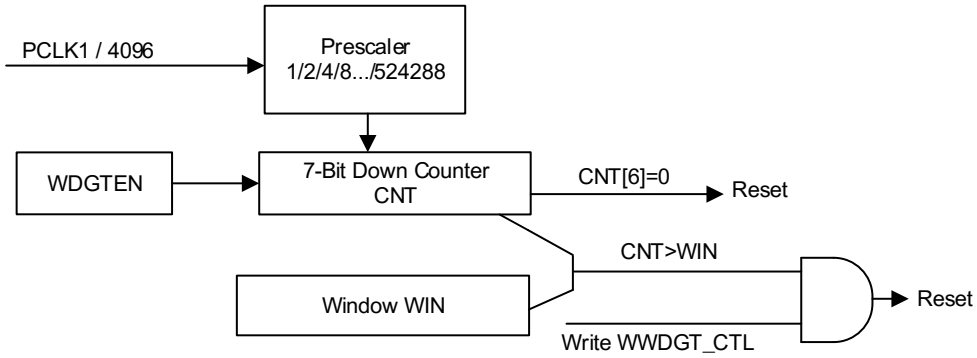
20.2.2. Characteristics

- Programmable free-running 7-bit downcounter.
- Generate a 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.

20.2.3. 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 becomes cleared), or the counter is refreshed before the counter reaches the window register value.

Figure 20-2. Window watchdog timer block diagram



The window watchdog timer is always disabled after power on reset. The software starts the

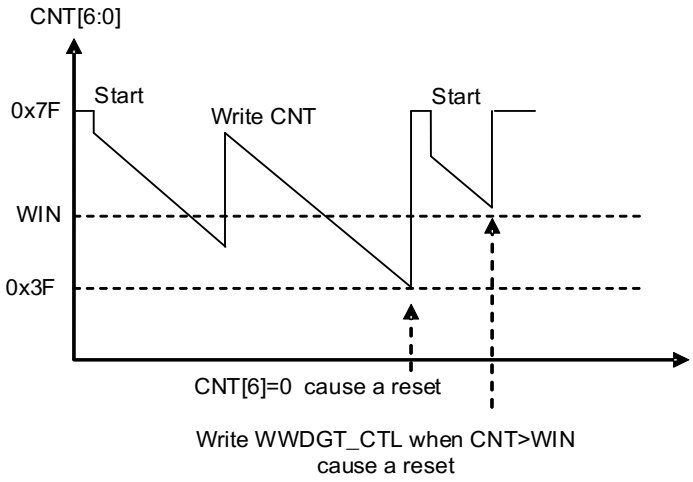
watchdog by setting the WDGTEN bit in the WWDGT_CTL register. When window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F, (it implies that the CNT[6] bit should be set). The CNT[5:0] determine the maximum time interval between two reloading. The count down speed depends on the APB1 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 20-3. Window watchdog timing diagram



Calculate the WWDGT timeout by using the formula below.

$$t_{WWDGT}=t_{PCLK1} \times 4096 \times 2^{PSC} \times (CNT[5:0]+1) \quad (ms) \qquad (20\text{-}1)$$

where:

t_{WWDGT}: WWDGT timeout

t_{PCLK1}: APB1 clock period measured in ms

The *Table 20-2. Min / max timeout value at 50 MHz (fPCLK1)* shows the minimum and maximum values of the t_{WWDGT}.

Table 20-2. Min / max timeout value at 50 MHz (f_{PCLK1})

Prescaler divider	PSC[4:0]	Min timeout value	Max timeout value

		CNT[6:0] =0x40	CNT[6:0]=0x7F
1 / 1	00000	0.082 ms	5.243 ms
1 / 2	00001	0.164 ms	10.486 ms
1 / 4	00010	0.328 ms	20.972 ms
1 / 8	00011	0.655 ms	41.943 ms
1 / 16	00100	1.311 ms	83.886 ms
1 / 32	00101	2.621 ms	167.772 ms
1 / 64	00110	5.243 ms	335.544 ms
1 / 128	00111	10.486 ms	671.089 ms
1 / 256	01000	20.972 ms	1342.177 ms
1 / 512	01001	41.943 ms	2684.355 ms
1 / 1024	01010	83.886 ms	5368.709 ms
1 / 2048	01011	167.772 ms	10737.418 ms
1 / 4096	01100	335.544 ms	21474.836 ms
1 / 8192	01101	671.089 ms	42949.673 ms
1 / 16384	01110	1342.177 ms	85899.346 ms
1 / 32768	01111	2684.355 ms	171798.692 ms
1 / 65536	10000	5368.709 ms	343597.384 ms
1 / 131072	10001	10737.418 ms	687194.767 ms
1 / 262144	10010	21474.836 ms	1374389.535 ms

If the WWDGT_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex®-M33 core halted (Debug mode). While the WWDGT_HOLD bit is set, the WWDGT stops in Debug mode.

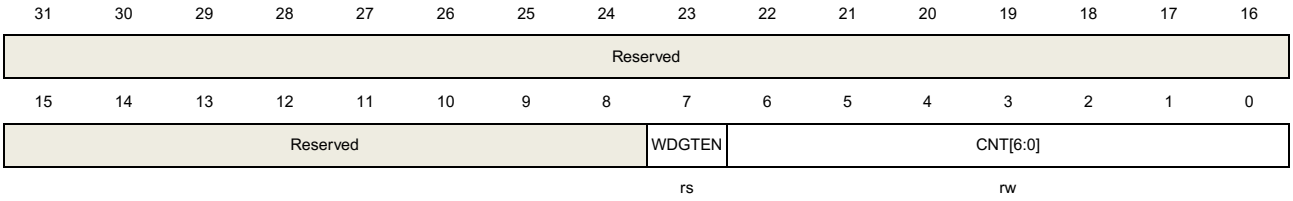
20.2.4. Register definition

WWDGT base address: 0x4000 2C00

Control register (WWDGT_CTL)

Address offset: 0x00
Reset value: 0x0000 007F

This register can be accessed by half-word (16-bit) or word (32-bit).

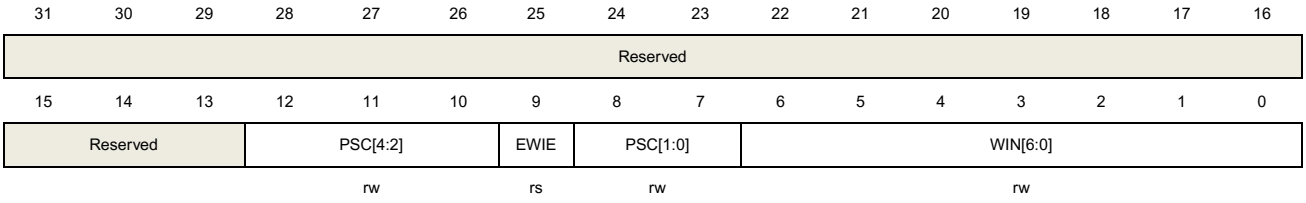


Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	WDGTEN	Start the window watchdog timer. It can be cleared by a hardware reset or software clock reset (refer to APB1 reset register (RCU_APB1RST)). Writing 0 has no effect. 0: Window watchdog timer disabled 1: Window watchdog timer enabled
6:0	CNT[6:0]	The value of the watchdog timer counter. A reset occurs 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 half-word (16-bit) or word (32-bit).



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:10	PSC[4:2]	High bit value of prescaler. PSC[4:0] consists of high bit value PSC[4:2] and low bit value PSC[1:0]. PSC[4:0] is the time base of the watchdog counter: 00000: (PCLK1 / 4096) / 1 00001: (PCLK1 / 4096) / 2

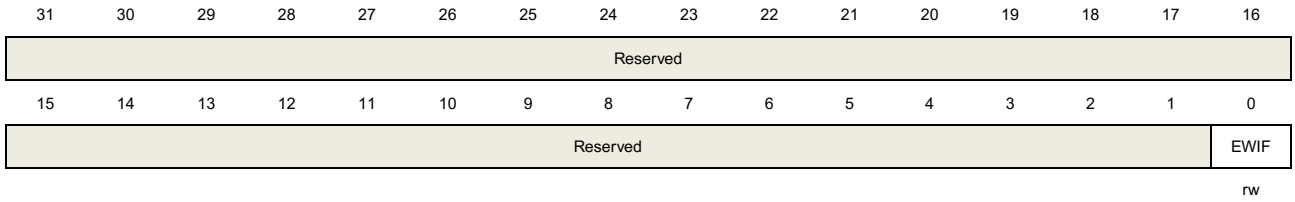
00010: (PCLK1 / 4096) / 4
00011: (PCLK1 / 4096) / 8
……
10001: (PCLK1 / 4096) / 131072
10010: (PCLK1 / 4096) / 262144
10100~11111: Reserved

9	EWIE	Early wakeup interrupt enable. An interrupt occurs when the counter reaches 0x40 if the bit is set. It can be cleared by a hardware reset or by a RCU WWDGT software reset. A write operation of '0' has no effect.
8:7	PSC[1:0]	Low bit value of prescaler. PSC[4:0] consists of high bit value PSC[4:2] and low bit value PSC[1:0]. PSC[4:0] is the time base of the watchdog counter.
6:0	WIN[6:0]	The Window value. A reset occurs 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 half-word (16-bit) or 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, 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 to it. There is no effect when writing 1 to it.

21. Real time clock (RTC)

21.1. 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.

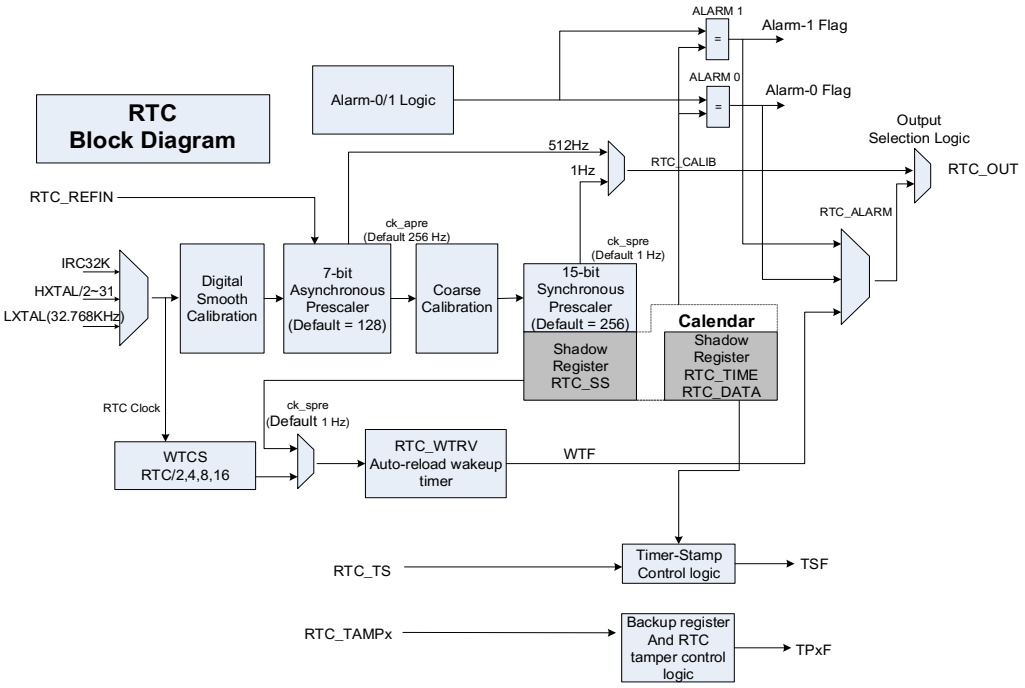
21.2. Characteristics

- Daylight saving compensation supported by 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
- Programmable calendar and two field maskable alarms
- Maskable interrupt source:
  - Alarm 0 and Alarm 1
  - Time-stamp detection
  - Tamper detection
  - Auto wakeup event
- Twenty 32-bit (80 bytes total) universal backup registers which can keep data under power saving mode. Backup register will be reset if tamper event detected

21.3. Function overview

21.3.1. Block diagram

Figure 21-1. Block diagram of RTC



The RTC unit includes:

- Alarm event/interrupt
- Tamper event/interrupt
- 32-bit backup registers
- Optional RTC output function:
  - 512Hz (default prescale) : (RTC_OUT)PC13
  - 1Hz(default prescale): (RTC_OUT)PC13
  - Alarm event(polarity is configurable): (RTC_OUT)PC13
  - Automatic wakeup event(polarity is configurable): (RTC_OUT)PC13
- Optional RTC input function:
  - time stamp event detection(RTC_TS): PC13 and PI8
  - tamper 0 event detection(RTC_TAMP0): PC13 and PI8
  - tamper 1 event detection(RTC_TAMP1): PI8
  - reference clock input RTC_REFIN(50 or 60 Hz)

PC13 and PI8 pin configuration refer to ***General-purpose and alternate-function I/Os (GPIO and AFIO)***

21.3.2. Clock source and prescalers

RTC unit has three independent clock sources: LXTAL, IRC32K and HXTAL with divided by

2~31(configured in RCU_CFG 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:

$$f_{ck_apre} = \frac{f_{rtcclk}}{FACTOR_A + 1} \quad (21\text{-}1)$$

$$f_{ck_spre} = \frac{f_{ck_apre}}{FACTOR_S + 1} = \frac{f_{rtcclk}}{(FACTOR_A + 1)*(FACTOR_S + 1)} \quad (21\text{-}2)$$

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.

21.3.3. 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 (fapb) must be at least 7 times the frequency of the RTC clock (frtcclk).

System reset will reset the shadow calendar registers.

21.3.4. 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 bit in RTC_CTL. If all the alarm fields value match the corresponding calendar value when ALRMxEN=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.

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 is set.

21.3.5. 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 PC13 from RTC_ALARM channel.

21.3.6. 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_COSC, RTC_ALRM0TD, RTC_ALRM1TD, RTC_SHIFTCTL, RTC_HRFC, RTC_ALRM0SS, RTC_ALRM1SS.

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 in RTC_CTL)
2. Set the Alarm registers needed(RTC_ALRMxTD/RTC_ALRMxSS)
3. Enable Alarm function (by setting ALRMxEN in the RTC_CTL)

21.3.7. 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 APB1 bus clock frequency must be equal to or greater than 7 times the RTC clock frequency.APB1 bus clock frequency lower than RTC clock frequency is not allowed in any case whatever happens.

When APB1 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.

21.3.8. 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 Coarse calibration register (RTC_COSC)
- 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 Alarm registers (RTC_ALRMxSS/RTC_ALRMxTD)

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.

21.3.9. 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.

21.3.10. 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 and must not be used with coarse digital function.

21.3.11. RTC coarse digital calibration

There are two digital methods can be chose for calibration: coarse digital calibration and smooth digital calibration. These two types cannot be used together.

Coarse digital calibration can be used to add or mask ck_apre clock cycles at the output of the asynchronous prescaler.

When COSD=0, 2 ck_apre cycles are added every minute for the first 2xCOSS minutes. The effect of such configuration will make calendar to be updated sooner.

When COSD=1, 1 ck_apre cycle is removed every minute for the first 2xCOSS minutes. The effect of such configuration will make calendar to be updated later.

Only in initialization mode can configure coarse calibration and the function starts after clearing INITM bit. The full calibration window lasts 64 minutes. The first 2xCOSS minutes of this 64-minute window are take adjust.

About 2PPM resolution is taken for negative calibration and about 4PPM resolution is taken for positive calibration.

Note: The calibration can be performed either on LXTAL or HXTAL clock. If FACTOR_A<6, the calibration may not work correctly.

Example:

FACTOR_A and FACTOR_S are default values. LXTAL is the RTC clock source and frequency is 32.768 KHz.

During a calibration window (64 minutes), the ck_apre clock frequency is only adjusted in the first 2xCOSS minutes. If COSS=1, this means only the first 2 minutes of 64 minutes will make adjustment.

The calibration step therefore has the effect of adding 512 or subtracting 256 oscillator cycles for each calibration window (64min x 60s/min x 32768cycles/s). In another word, this is equivalent to +4.069PPM or -2.035PPM per calibration step. Then for one month running, the minimum calibration step is +10.5 or -5.27 seconds and the maximum calibration step is +5.45 to -2.72 minutes.

21.3.12. 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 220/219/218 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:

$$f_{cal}=f_{rtcclk} \times \left(1+\frac{FREQI\times 512-CMSK}{2^{N}+CMSK-FREQI\times 512}\right) \quad (21\text{-}3)$$

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:

$$f_{cal}=f_{rtcclk} \times \left(1+\frac{256-CMSK}{2^{N}+CMSK-256}\right) \quad (21\text{-}4)$$

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

21.3.13. Time-stamp function

Time-stamp function is performed on RTC_TS pin and is enabled by control bit TSEN.

When a time-stamp event occurs on RTC_TS pin, 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.

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.

21.3.14. 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.

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. When the tamper event is detected, the corresponding flag (TPxF) will assert. Tamper event can generate an interrupt if tamper interrupt enable (TPIE) is set. Any tamper event will reset all backup registers (RTC_BKPx).

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.

21.3.15. Calibration clock output

Calibration clock can be output on the PC13 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 frtcclk/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:

$$f_{rtc_calib} = \frac{f_{rtcclk}}{(FACTOR_A+1)\times(FACTOR_S+1)} \quad (21\text{-}5)$$

When the RTCCLK is 32.768 KHz, RTC_CALIB output is corresponding to 1Hz if prescaler

are default values.

21.3.16. 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.

21.3.17. RTC power saving mode management

Table 21-1 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

21.3.18. 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 21-2 RTC interrupts control

Interrupt	Event flag	Control Bit	Exit Sleep	Exit Deep-sleep	Exit Standby
Alarm 0	ALRM0F	ALRM0IE	Y	Y(*)	Y(*)
Alarm 1	ALRM1F	ALRM1IE	Y	Y(*)	Y(*)
Wakeup	WTF	WTIE	Y	Y(*)	Y(*)
Timestamp	TSF	TSIE	Y	Y(*)	Y(*)
Tamper 0	TP0F	TPIE	Y	Y(*)	Y(*)
Tamper 1	TP1F	TPIE	Y	Y(*)	Y(*)

NOTE: (*) Only active when RTC clock source is LXTAL or IRC32K.

21.4. Register definition

RTC base address: 0x4000 2800

21.4.1. Time register (RTC_TIME)

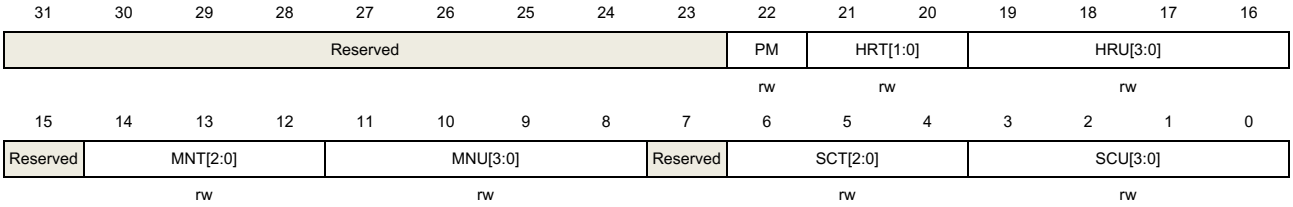
Address offset: 0x00

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 0: AM or 24-hour format 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

21.4.2. 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)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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 0x0: Reserved 0x1: Monday … 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

21.4.3. 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								COEN	OS[1:0]		OPOL	COS	DSM	S1H	A1H
								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	CCEN	CS	BPSHAD	REFEN	TSEG	WTCS[2:0]		
rw	rw	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	COEN	Calibration output enable 0: Disable calibration output 1: Enable calibration output

22:21	OS[1:0]	Output selection This bit is used for selecting flag source to output 0x0: Disable output RTC_ALARM 0x1: Enable alarm0 flag output 0x2: Enable alarm1 flag output 0x3: Enable wakeup flag output
20	OPOL	Output polarity This bit is used to invert output RTC_ALARM 0: Disable invert output RTC_ALARM 1: Enable invert output RTC_ALARM
19	COS	Calibration output selection Valid only when COEN=1 and prescalers are at default values 0: Calibration output is 512 Hz 1: Calibration output is 1Hz
18	DSM	Daylight saving mark 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) One hour will be subtracted from current time if it is not 0 0: No effect 1: 1 hour will be subtracted at next second change time.
16	A1H	Add 1 hour(summer time change) One hour will be added from current time 0: No effect 1: 1 hour will be added at next second change time
15	TSIE	Time-stamp interrupt enable 0: Disable time-stamp interrupt 1: Enable time-stamp interrupt
14	WTIE	Auto-wakeup timer interrupt enable 0: Disable auto-wakeup timer interrupt 1: Enable auto-wakeup timer interrupt
13	ALRM1IE	RTC alarm-1 interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
12	ALRM0IE	RTC alarm-0 interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
11	TSEN	Time-stamp function enable 0: Disable time-stamp function

		1: Enable time-stamp function
10	WTEN	Auto-wakeup timer function enable 0: Disable function 1: Enable function
9	ALRM1EN	Alarm-1 function enable 0: Disable alarm function 1: Enable alarm function
8	ALRM0EN	Alarm-0 function enable 0: Disable alarm function 1: Enable alarm function
7	CCEN	Coarse calibration function enable 0: Disable function 1: Enable function Note: FACTOR_A must be greater than 6 before enabled and can only be written in initialization state.
6	CS	Clock System 0: 24-hour format 1: 12-hour format Note: Can only be written in initialization state
5	BPSHAD	Shadow registers bypass control 0: Reading calendar from shadow registers 1: Reading calendar from current real-time calendar Note: If frequency of APB1 clock is less than seven times the frequency of RTCCLK, this bit must set to 1.
4	REFEN	Reference clock detection function enable 0: Disable reference clock detection function 1: Enable reference clock detection function Note: Can only be written in initialization state and FACTOR_S must be 0x00FF
3	TSEG	Valid event edge of time-stamp 0: rising edge is valid event edge for time-stamp event 1: falling edge is valid event edge for time-stamp event
2:0	WTCS[2:0]	Auto-wakeup timer clock selection 0x0:RTC Clock divided by 16 0x1:RTC Clock divided by 8 0x2:RTC Clock divided by 4 0x3:RTC Clock divided by 2 0x4:0x5: ck_spre (default 1Hz) clock 0x6:0x7: ck_spre (default 1Hz) clock and 2^{16} is added to wake-up counter.

21.4.4. 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[14:8].

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															SCPF
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TP1F	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
31:17	Reserved	Must be kept at reset value.
16	SCPF	Smooth calibration pending flag Set to 1 by hardware when software writes to RTC_HRFC without entering initialization mode and set to 0 by hardware when smooth calibration configuration is taken into account.
15	Reserved	Must be kept at reset value.
14	TP1F	RTC_TAMP1 detected flag 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.
13	TP0F	RTC_TAMP0 detected flag 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 This bit is set by hardware when a time-stamp event is detected if TSF bit is set before. Cleared by software writing 0.
11	TSF	Time-stamp flag Set by hardware when time-stamp event is detected. Cleared by software writing 0.
10	WTF	Wakeup timer flag Set by hardware when wakeup timer decreased to 0. Cleared by software writing 0. 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. Cleared by software writing 0.
8	ALRM0F	Alarm-0 occurs flag Set to 1 by hardware when current time/date matches the time/date of alarm 0 setting value. Cleared by software writing 0.
7	INITM	Enter initialization mode 0: Free running mode 1: Enter initialization mode for setting calendar time/date and prescaler. Counter will stop under this mode.
6	INITF	Initialization state flag Set to 1 by hardware and calendar register and prescaler can be programmed in this state. 0: Calendar registers and prescaler register cannot be changed 1: Calendar registers and prescaler register can be changed
5	RSYNF	Register synchronization flag 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. 0:Shadow register are not yet synchronized 1:Shadow register are synchronized
4	YCM	Year configuration mark Set by hardware if the year field of calendar date register is not the default value 0. 0: Calendar has not been initialized 1: Calendar has been initialized
3	SOPF	Shift function operation pending flag 0: No shift operation is pending 1: Shift function operation is pending
2	WTWF	Wakeup timer write enable flag 0: Wakeup timer update is not allowed 1: Wakeup timer update is allowed
1	ALRM1WF	Alarm 1 configuration can be write flag Set by hardware if alarm register can be wrote after ALRM1EN bit has reset. 0: Alarm registers programming is not allowed 1: Alarm registers programming is allowed
0	ALRM0WF	Alarm 0 configuration can be write flag 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.

21.4.5. Prescaler register (RTC_PSC)

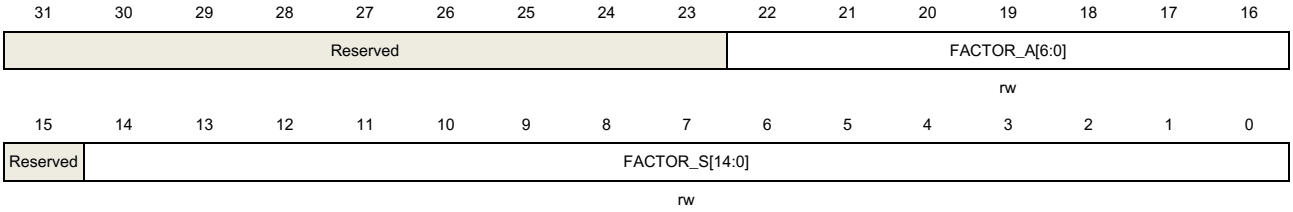
Address offset: 0x10

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 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 ck_spre frequency = ck_apre frequency/(FACTOR_S+1)

21.4.6. Wakeup timer register (RTC_WUT)

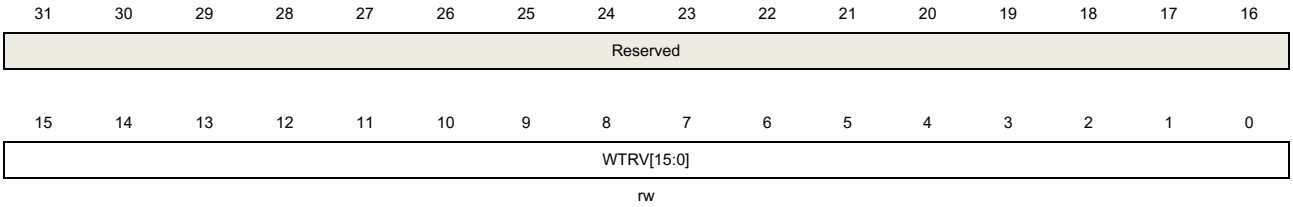
Address offset: 0x14

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)



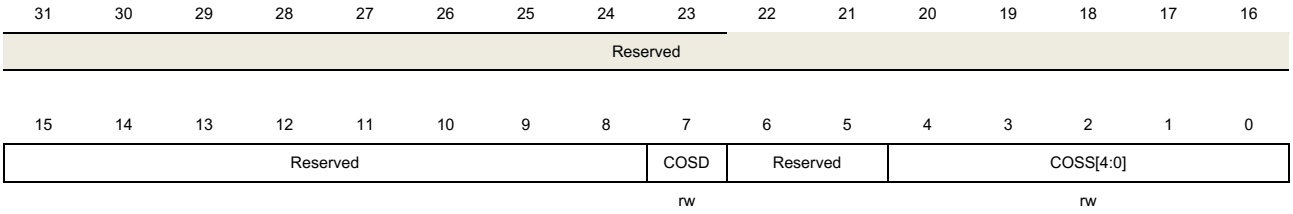
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	WTRV[15:0]	Auto-wakeup timer reloads value. Every (WTRV[15:0]+1) ck_wut period the WTF bit is set after WTEN=1.The ck_wut is selected by WTCS[2:0] bits.

Note: This configure case is forbidden: WTRV=0x0000 with WTCS[2:0]=0b011.
This register can be written only when WTWF=1.

21.4.7. Coarse calibration register (RTC_COSC)

Address offset: 0x18
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)



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	COSD	Coarse Calibration direction 0: Increase calendar update frequency 1: Decrease calendar update frequency
6:5	Reserved	Must be kept at reset value.
4:0	COSS[4:0]	Coarse Calibration step When COSD=0: 0x00:+0 PPM 0x01:+4 PPM(approximate value) 0x02:+8 PPM(approximate value) .... 0x1F:+126 PPM(approximate value) When COSD=1: 0x00:-0 PPM 0x01:-2 PPM(approximate value) 0x02:-4 PPM(approximate value) … 0x1F:-63 PPM(approximate value)

21.4.8. 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 0: Not mask date/day field 1: Mask date/day field
30	DOWS	Day of the week selected 0: DAYU[3:0] indicates the date units 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 0: Not mask hour field 1: Mask hour field
22	PM	AM/PM flag 0: AM or 24-hour format 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 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 0: Not mask second field 1: Mask second field
6:4	SCT[2:0]	Second tens in BCD code
3:0	SCU[3:0]	Second units in BCD code

21.4.9. 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			

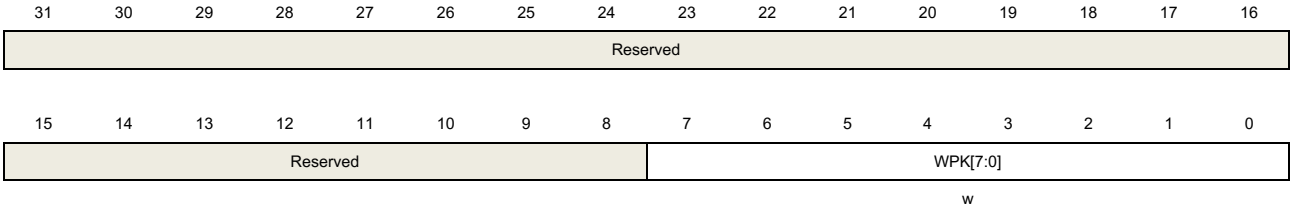
Bits	Fields	Descriptions
31	MSKD	Alarm date mask bit 0: Not mask date/day field 1: Mask date/day field
30	DOWS	Day of the week selected 0: DAYU[3:0] indicates the date units 1: DAYU[3:0] indicates the week day and DAYT[3:0] has no means.
29:28	DAYT[1:0]	Day tens in BCD code
27:24	DAYU[3:0]	Day units or week day in BCD code
23	MSKH	Alarm hour mask bit 0: Not mask hour field 1: Mask hour field
22	PM	AM/PM flag 0: AM or 24-hour format 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 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 0: Not mask second field 1: Mask second field
6:4	SCT[2:0]	Second tens in BCD code
3:0	SCU[3:0]	Second units in BCD code

21.4.10. 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

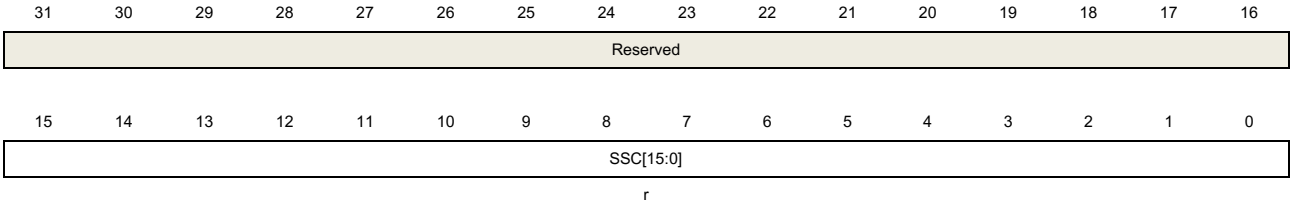
21.4.11. Sub second register (RTC_SS)

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)

21.4.12. 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

Note: Writing to this register will cause RSYNF bit to be cleared.

This register 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 0: Not add 1 second 1: Add 1 second to the clock/calendar. 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 The value of this bit will add to the counter of synchronous prescaler. When only using SFS, the clock will delay because the synchronous prescaler is a down counter: Delay (seconds) = SFS / (FACTOR_S + 1) When jointly using A1S and SFS, the clock will advance: Advance (seconds) = (1 - (SFS / (FACTOR_S + 1)))

21.4.13. 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]			
									r	r		r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MNT[2:0]			MNU[3:0]				Reserved	SCT[2:0]			SCU[3:0]			
	r			r					r			r			

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22	PM	AM/PM mark 0:AM or 24-hour format 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

21.4.14. 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)

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
DOW[2:0]			MONT	MONU[3:0]				Reserved		DAYT[1:0]		DAYU[3:0]			
r			r	r				r				r			

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

21.4.15. 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)

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
SSC[15:0]															
r															

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 when TSF is set to 1.

21.4.16. 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)

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
FREQI	CWND8	CWND16	Reserved								CMSK[8:0]				
rw	rw	rw									rw				

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	FREQI	Increase RTC frequency by 488.5PPM 0: No effect 1: One RTCCLK pulse is inserted every 2^{11} pulses. 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 0: No effect 1: Calibration window is 8 second Note: When CWND8=1, CMSK[1:0] are stuck at "00".
13	CWND16	Frequency compensation window 16 second selected 0: No effect 1: Calibration window is 16 second

		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 The number of mask pulse out of 2^{20} RTCCLK pulse. This feature will decrease the frequency of calendar with a resolution of 0.9537 PPM.

21.4.17. 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													AOT	TSSEL	TP0SEL
													rw	rw	rw

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	Reserved		TP1EG	TP1EN	TPIE	TP0EG	TP0EN
rw	rw		rw		rw			rw			rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value.
18	AOT	RTC_ALARM Output Type 0: Open-drain output type 1: Push-pull output type
17	TSSEL	Timestamp input selection: 0: Timestamp function input from PC13 1: Timestamp function input from PI8
16	TP0SEL	Tamper 0 function input selection 0: Tamper 0 function input from PC13 1: Tamper 0 function input from PI8. **Note:** TP0EN must be reset when TP0SEL is changed
15	DISPU	RTC_TAMPx pull up disable bit 0: Enable inner pull-up before sampling for pre-charge RTC_TAMPx pin 1: Disable pre-charge duration
14:13	PRCH[1:0]	Pre-charge duration time of RTC_TAMPx This setting determines the pre-charge time before each sampling. 0x0: 1 RTC clock 0x1: 2 RTC clock 0x2: 4 RTC clock 0x3: 8 RTC clock

12:11	FLT[1:0]	RTC_TAMPx filter count setting This bit determines the tamper sampling type and the number of consecutive sample. 0x0: Detecting tamper event using edge mode. Pre-charge duration is disabled automatically 0x1: Detecting tamper event using level mode.2 consecutive valid level samples will make an effective tamper event 0x2: Detecting tamper event using level mode.4 consecutive valid level samples will make an effective tamper event 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 0x0: Sample once every 32768 RTCCLK(1Hz if RTCCLK=32.768KHz) 0x1: Sample once every 16384 RTCCLK(2Hz if RTCCLK=32.768KHz) 0x2: Sample once every 8192 RTCCLK(4Hz if RTCCLK=32.768KHz) 0x3: Sample once every 4096 RTCCLK(8Hz if RTCCLK=32.768KHz) 0x4: Sample once every 2048 RTCCLK(16Hz if RTCCLK=32.768KHz) 0x5: Sample once every 1024 RTCCLK(32Hz if RTCCLK=32.768KHz) 0x6: Sample once every 512 RTCCLK(64Hz if RTCCLK=32.768KHz) 0x7: Sample once every 256 RTCCLK(128Hz if RTCCLK=32.768KHz)
7	TPTS	Make tamper function used for timestamp function 0:No effect 1:TSF is set when tamper event detected even TSEN=0
6:5	Reserved	Must be kept at reset value.
4	TP1EG	Tamper 1 event trigger edge 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
3	TP1EN	Tamper 1 detection enable 0: Disable tamper 1 detection function 1: Enable tamper 1 detection function
2	TPIE	Tamper detection interrupt enable 0: Disable tamper interrupt 1: Enable tamper interrupt
1	TP0EG	Tamper 0 event trigger edge 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

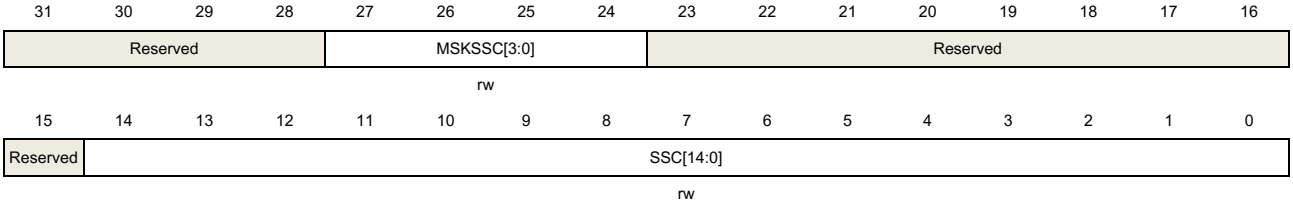
		If tamper detection is in level mode(FLT !=0): 0: Low level triggers a tamper detection event 1: High level triggers a tamper detection event
0	TP0EN	Tamper 0 detection enable 0:Disable tamper 0 detection function 1:Enable tamper 0 detection function

Note: It's strongly recommended that reset the TPxEN before change the tamper configuration.

21.4.18. 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 0x0: Mask alarm sub second setting. The alarm asserts at every second time point if all the rest alarm fields are matched. 0x1: SSC[0] is to be compared and all others are ignored 0x2: SSC[1:0] is to be compared and all others are ignored 0x3: SSC[2:0] is to be compared and all others are ignored 0x4: SSC[3:0] is to be compared and all others are ignored 0x5: SSC[4:0] is to be compared and all others are ignored 0x6: SSC[5:0] is to be compared and all others are ignored 0x7: SSC[6:0] is to be compared and all others are ignored 0x8: SSC[7:0] is to be compared and all others are ignored 0x9: SSC[8:0] is to be compared and all others are ignored 0xA: SSC[9:0] is to be compared and all others are ignored 0xB: SSC[10:0] is to be compared and all others are ignored 0xC: SSC[11:0] is to be compared and all others are ignored 0xD: SSC[12:0] is to be compared and all others are ignored 0xE: SSC[13:0] is to be compared and all others are ignored 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 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.

21.4.19. Alarm 1 sub second register (RTC_ALRM1SS)

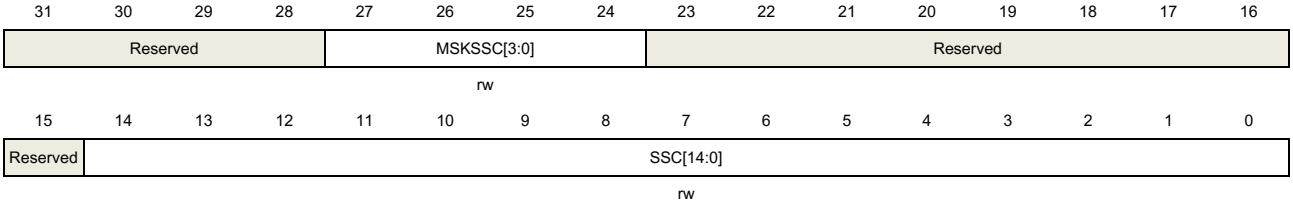
Address offset: 0x48

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 0x0: Mask alarm sub second setting. The alarm asserts at every second time point if all the rest alarm fields are matched. 0x1: SSC[0] is to be compared and all others are ignored 0x2: SSC[1:0] is to be compared and all others are ignored 0x3: SSC[2:0] is to be compared and all others are ignored 0x4: SSC[3:0] is to be compared and all others are ignored 0x5: SSC[4:0] is to be compared and all others are ignored 0x6: SSC[5:0] is to be compared and all others are ignored 0x7: SSC[6:0] is to be compared and all others are ignored 0x8: SSC[7:0] is to be compared and all others are ignored 0x9: SSC[8:0] is to be compared and all others are ignored 0xA: SSC[9:0] is to be compared and all others are ignored 0xB: SSC[10:0] is to be compared and all others are ignored 0xC: SSC[11:0] is to be compared and all others are ignored 0xD: SSC[12:0] is to be compared and all others are ignored 0xE: SSC[13:0] is to be compared and all others are ignored 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 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.

21.4.20. Backup registers (RTC_BKPx) (x=0..19)

Address offset: 0x50~0x9C

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[31:0]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[31:0]															
rw															

Bits	Fields	Descriptions
31:0	DATA[31:0]	Data 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.

22. Timer (TIMERx)

Table 22-1. Timers (TIMERx) are divided into five sorts

TIMER	TIMER0 / 7	TIMER1~4	TIMER8 / 11	TIMER9 / 10 / 12 / 13	TIMER5 / 6
TYPE	Advanced	General-L0	General-L1	General-L2	Basic
Prescaler	16-bit	16-bit	16-bit	16-bit	16-bit
Counter	16-bit	32-bit(TIMER1&4) 16-bit(TIMER2&3)	16-bit	16-bit	16-bit
Count mode	Up,Down, Center-aligned	Up,Down, Center-aligned	Up Only	Up Only	Up Only
Repetition	●	×	×	×	×
CH Capture/ Compare	4	4	2	1	0
Composite PWM mode	●	×	×	×	×
Complementary & Dead-time	●	×	×	×	×
Break	●	×	×	×	×
Single Pulse	●	●	●	×	●
Quadrature Decoder	●	●	×	×	×
Master-slave management	●	●	●	×	×
Inter connection	●(1)	●(2)	●(3)	×	Trgo to DAC
DMA	●	●	×	×	●(4)
Debug Mode	●	●	●	●	●

(1) TIMER0 **ITI0:** TIMER4_TRGO **ITI1:** TIMER1_TRGO **ITI2:** TIMER2_TRGO **ITI3:** TIMER3_TRGO
TIMER7 **ITI0:** TIMER0_TRGO **ITI1:** TIMER1_TRGO **ITI2:** TIMER3_TRGO **ITI3:** TIMER4_TRGO

(2) TIMER1 **ITI0:** TIMER0_TRGO **ITI1:** TIMER7_TRGO **ITI2:** TIMER2_TRGO **ITI3:** TIMER3_TRGO
TIMER2 **ITI0:** TIMER0_TRGO **ITI1:** TIMER1_TRGO **ITI2:** TIMER4_TRGO **ITI3:** TIMER3_TRGO
TIMER3 **ITI0:** TIMER0_TRGO **ITI1:** TIMER1_TRGO **ITI2:** TIMER2_TRGO **ITI3:** TIMER7_TRGO
TIMER4 **ITI0:** TIMER1_TRGO **ITI1:** TIMER2_TRGO **ITI2:** TIMER3_TRGO **ITI3:** TIMER7_TRGO

(3) TIMER8 **ITI0:** TIMER1_TRGO **ITI1:** TIMER2_TRGO **ITI2:** TIMER9_TRGO **ITI3:** TIMER10_ TRGO
TIMER11 **ITI0:** TIMER3_TRGO **ITI1:** TIMER4_TRGO **ITI2:** TIMER12_TRGO **ITI3:** TIMER13_ TRGO

(4) Only update events will generate DMA request. Note that TIMER5/6 do not have DMA configuration registers.

22.1. Advanced timer (TIMERx, x=0, 7)

22.1.1. Overview

The advanced timer module (Timer0&Timer7) is a four-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 issuitable for motor control applications.

Timer and timer are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

22.1.2. Characteristics

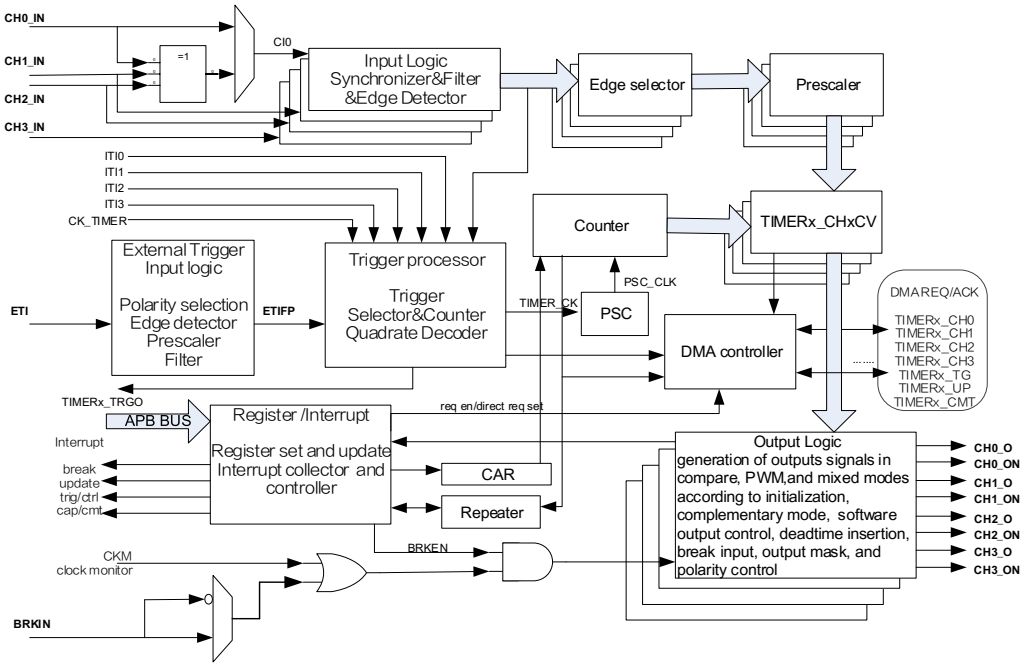
- Total channel num: 4.
- Counter width: 16 bit.
- Source of counter clock is selectable:
  internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature Decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- 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.
- Interrupt output or DMA request on: update, trigger event, compare/capture event, commutation 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.

22.1.3. Function overview

Block diagram

Figure 22-1. Advanced timer block diagram provides details of the internal configuration of the advanced timer.

Figure 22-1. Advanced timer block diagram



Clock source configuration

The advanced timer has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

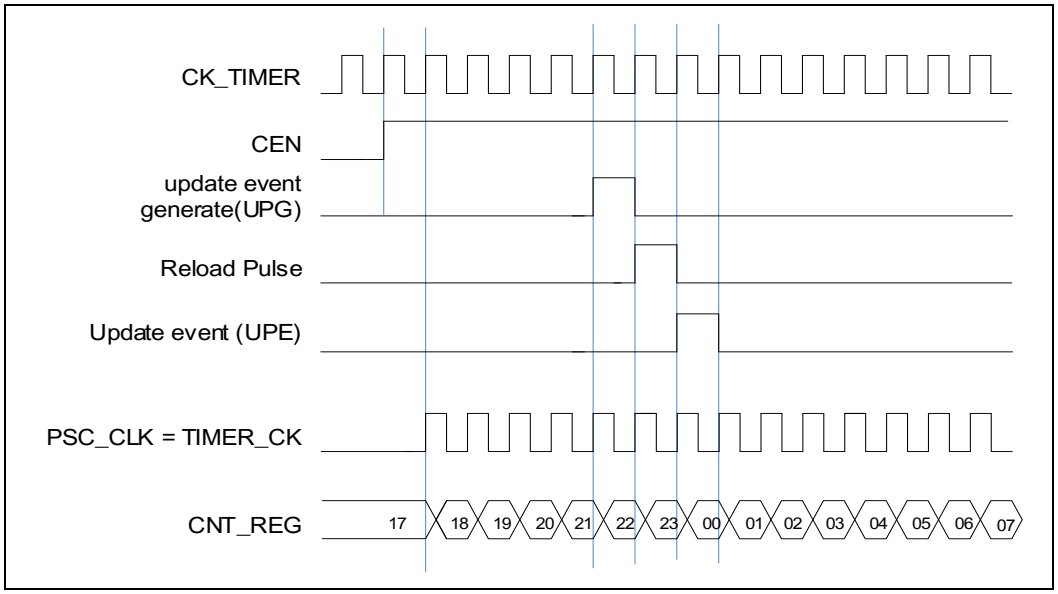
- SMC [2:0] == 3'b000. Internal clock CK_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK_TIMER for driving the counter prescaler when the SMC [2:0] == 3'b000. 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 the SMC [2:0] in the TIMERx_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx_SMCFG register, details as follows. When SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

Figure 22-2. Timing chart of internal clock divided by 1



- SMC [2:0] == 3'b111 (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 SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

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 SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

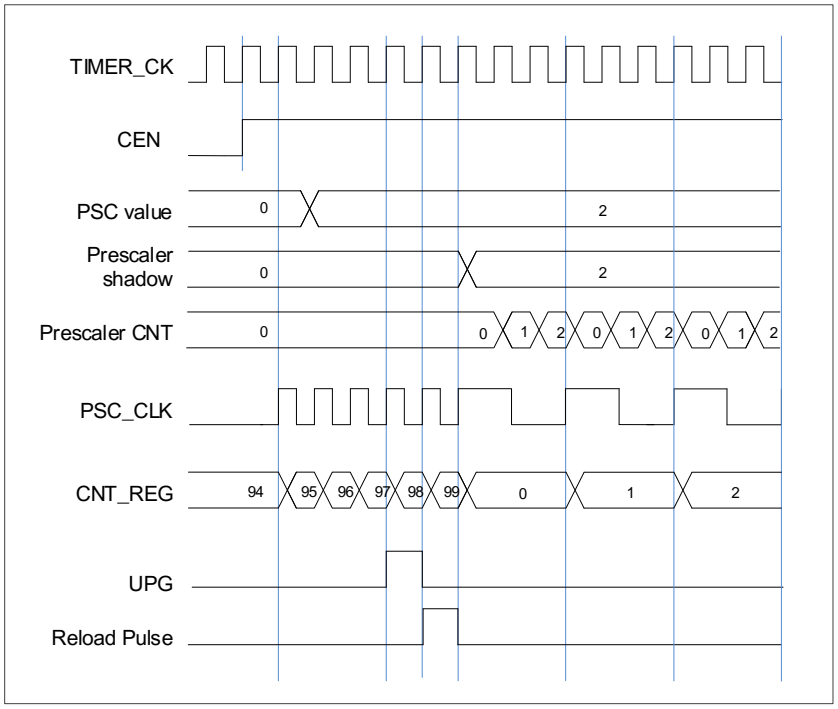
- SMC1== 1'b1 (external clock mode 1). External input is selected as timer clock source (ETI)

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 to set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as 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.

Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 22-3. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again and an overflow event will be generated. In addition, the update events will be generated after (TIMERx_CREP+1) times of overflow events. 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 shadow registers (repetition counter, auto reload register, prescaler register) are updated.

Figure 22-4. Timing chart of up counting mode, PSC=0/2 show some examples of the

counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

Figure 22-4. Timing chart of up counting mode, PSC=0/2

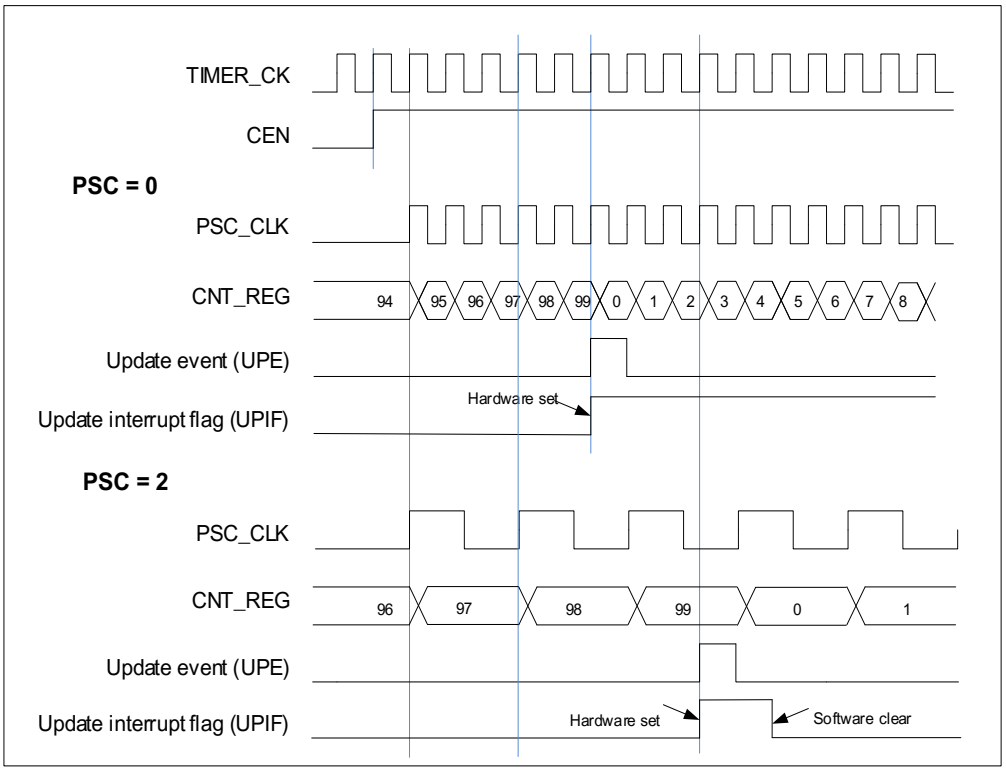
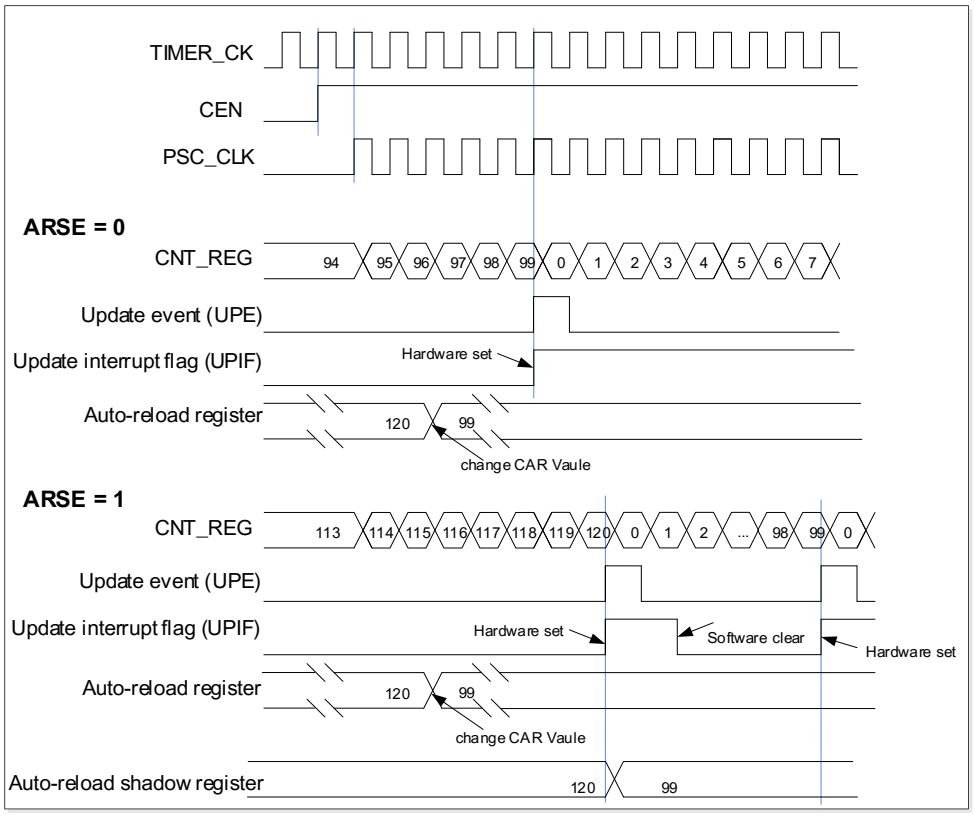


Figure 22-5. Timing chart of up counting mode, change TIMERx_CAR on the go



Counter down counting

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter the counter will start counting down from the counter-reload value again and an underflow event will be generated. In addition, the update event will be generated after (TIMERx_CREP+1) times of underflow. 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 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 shadow registers (repetition counter, auto reload register, prescaler register) are updated.

Figure 22-6. Timing chart of down counting mode, PSC=0/2 show some examples of the

counter behavior in different clock frequencies when TIMERx_CAR=0x99.

Figure 22-6. Timing chart of down counting mode, PSC=0/2

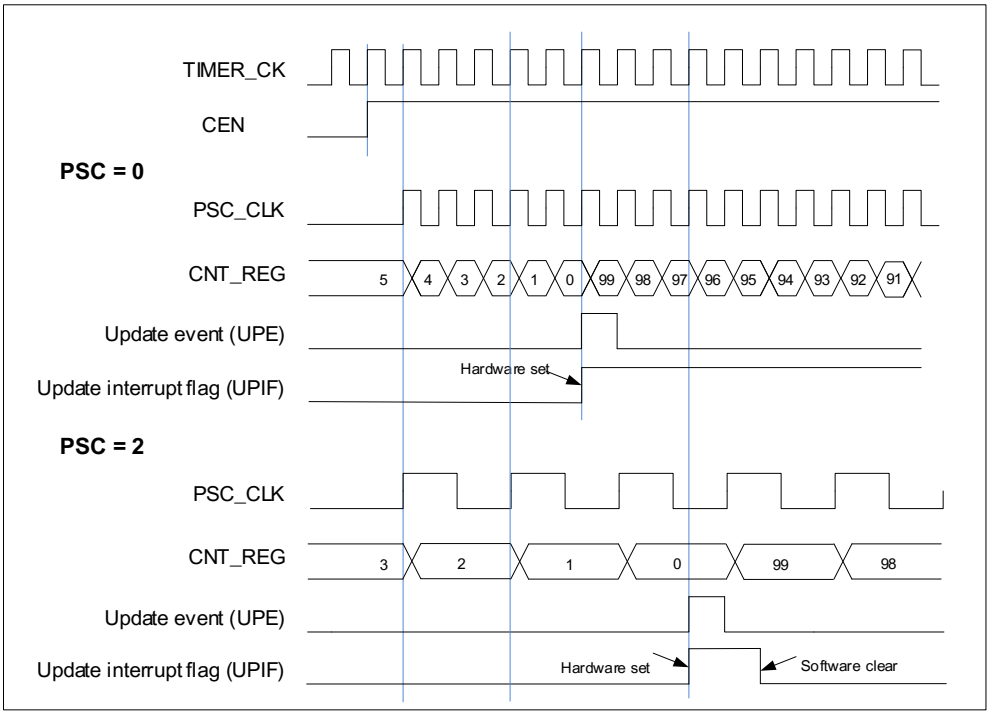
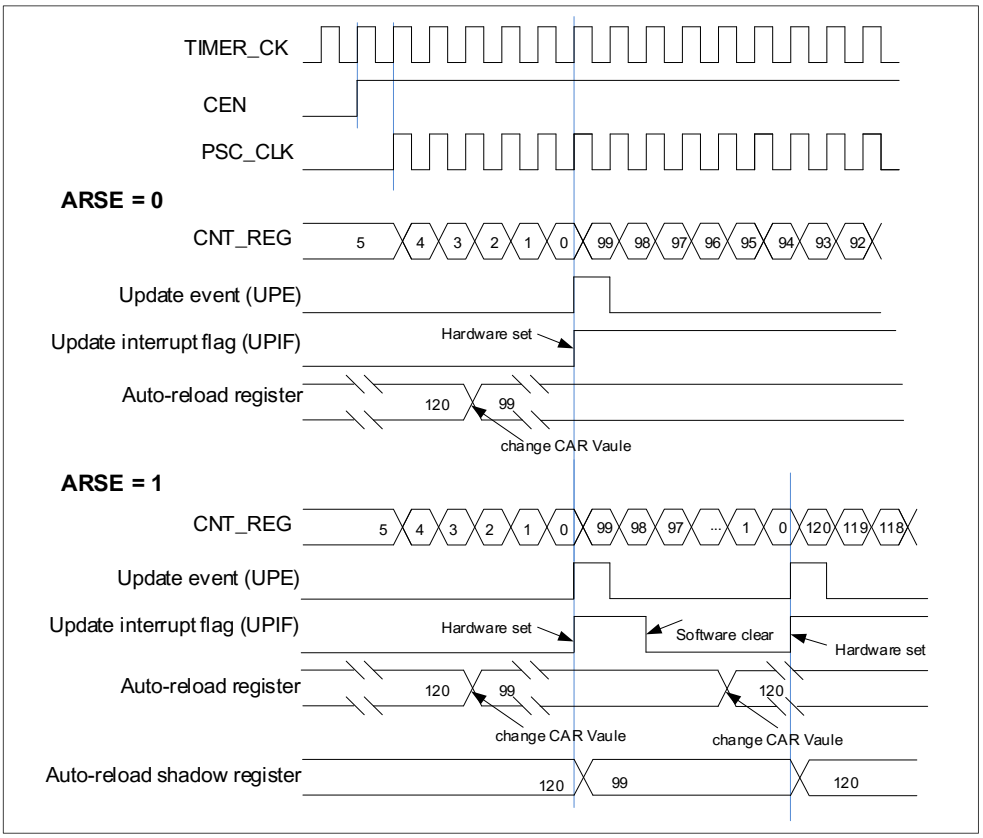


Figure 22-7. Timing chart of down counting mode, change TIMERx_CAR on the go



Counter center-aligned counting

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 the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting direction. The counting direction bit DIR in the TIMERx_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

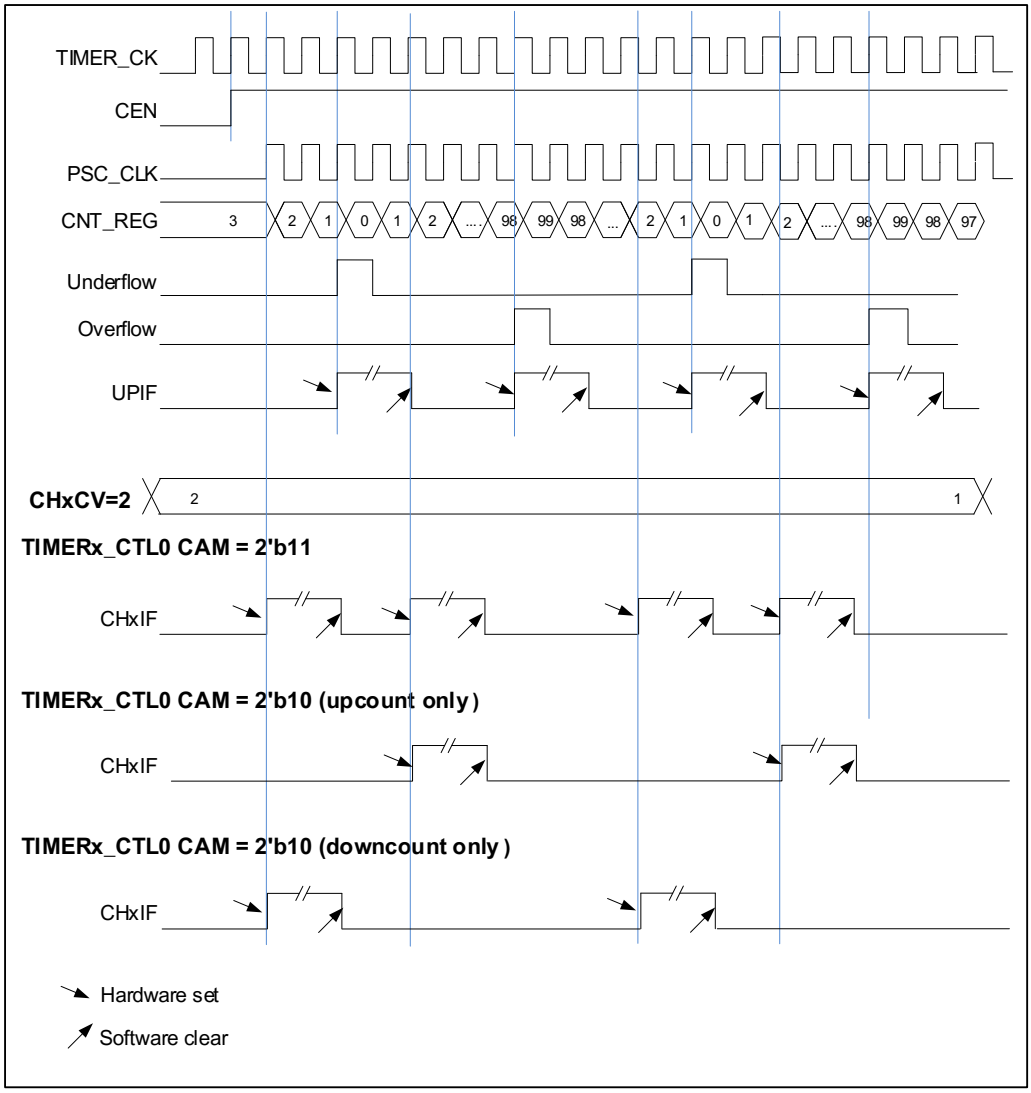
The UPIF bit in the TIMERx_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx_CTL0. The details refer to ***Figure 22-8. Center-aligned counter timechart***.

If set the UPDIS bit in the TIMERx_CTL0 register, the update event is disabled.

When an update event occurs, all the shadow registers (repetition counter, auto-reload register, prescaler register) are updated.

Figure 22-8. Center-aligned counter timechart show some examples of the counter behavior when TIMERx_CAR=0x99. TIMERx_PSC=0x0

Figure 22-8. Center-aligned counter timechart



Update event (from overflow/underflow) rate configuration

The rate of update events generation (from overflow and underflow events) can be configured by the TIMERx_CREP register. 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 CREP in TIMERx_CREP register. The repetition counter is decremented at each counter overflow (does not exist in down counting mode) and underflow (does not exist in up counting mode).

Setting the UPG bit in the TIMERx_SWEVG register will reload the content of CREP in TIMERx_CREP register and generator an update event.

The new written CREP value will not take effect until the next update event. When the value of CREP 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 CREP value takes effect. If an update event is generated by software after writing an odd number to CREP, the update

events will be generated on the underflow. If the next update event occurs on overflow after writing an odd number to CREP, then the subsequent update events will be generated on the overflow.

Figure 22-9. Repetition timechart for center-aligned counter

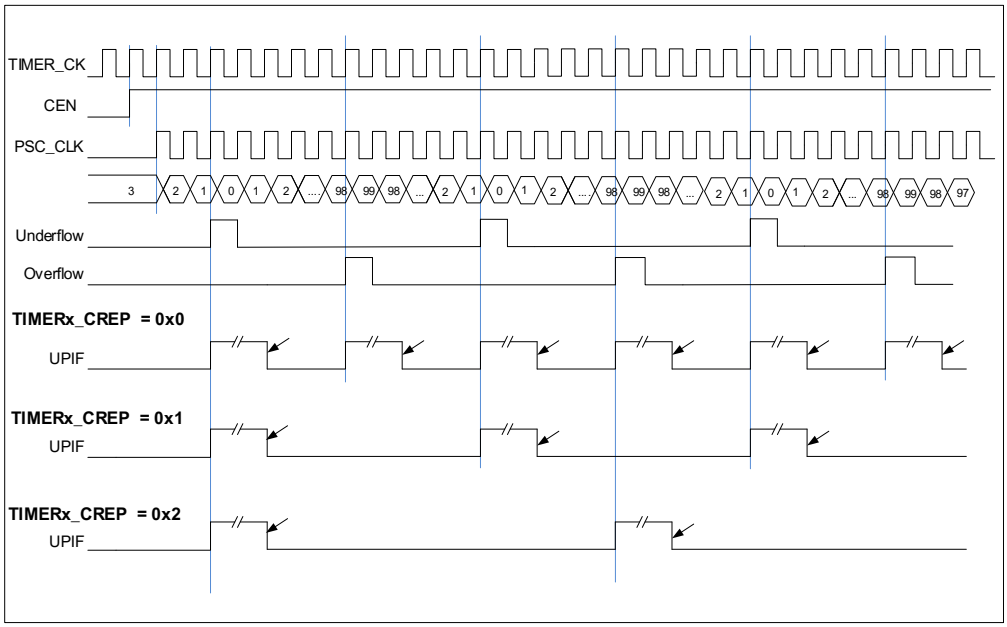


Figure 22-10. Repetition timechart for up-counter

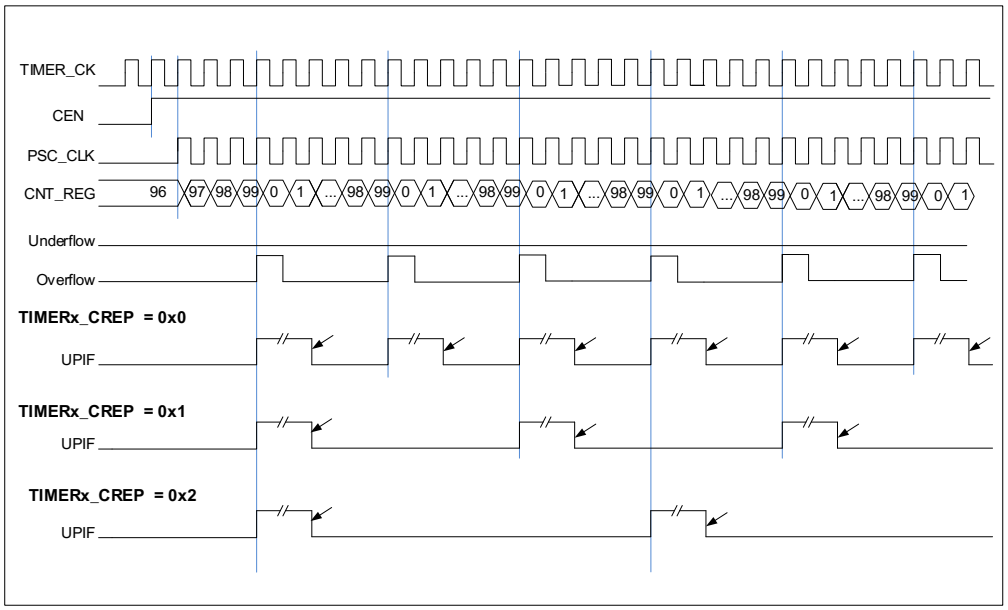
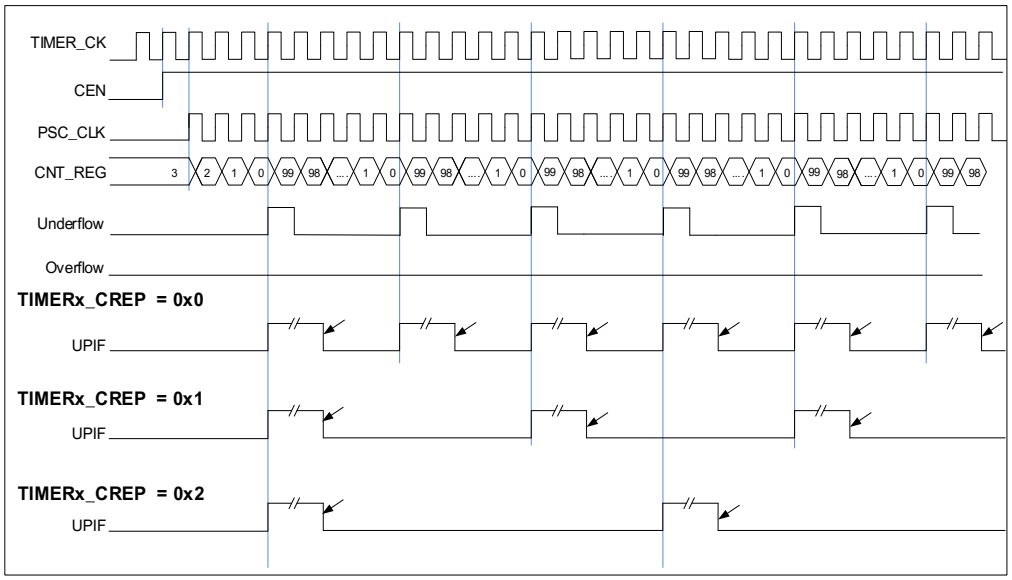


Figure 22-11. Repetition timechart for down-counter



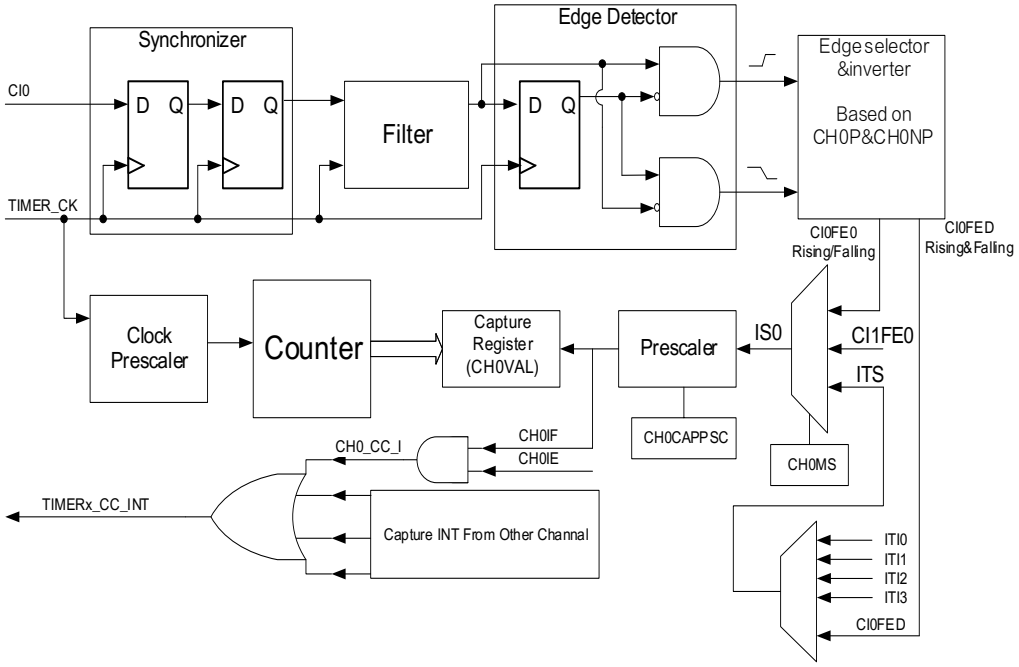
Input capture and output compare channels

The advanced 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.

- **Channel input capture function**

Channel input capture function 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 enabled by CHxIE = 1.

Figure 22-12. Channel input capture principle



One of channels' input signals (CIx) can be chosen from the TIMERx_CHx signal or the Excusive-OR function of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. First, the channel input signal (CIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generates one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

Step1: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

Step2: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source selection. (CHxMS in TIMERx_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel 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 enable; you can got the interrupt and DMA request.

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: when you wanted input signal is got, TIMERx_CHxCV will be set by counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt

and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN.

Direct generation: if you want to generate an Interrupt or DMA request, you can set CHxG by software directly.

The channel input capture function can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

■ Channel output compare function

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL 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 CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=1.

So the process can be divided to 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 high polarity by CHxP/CHxNP
* Enable the output by CHxEN

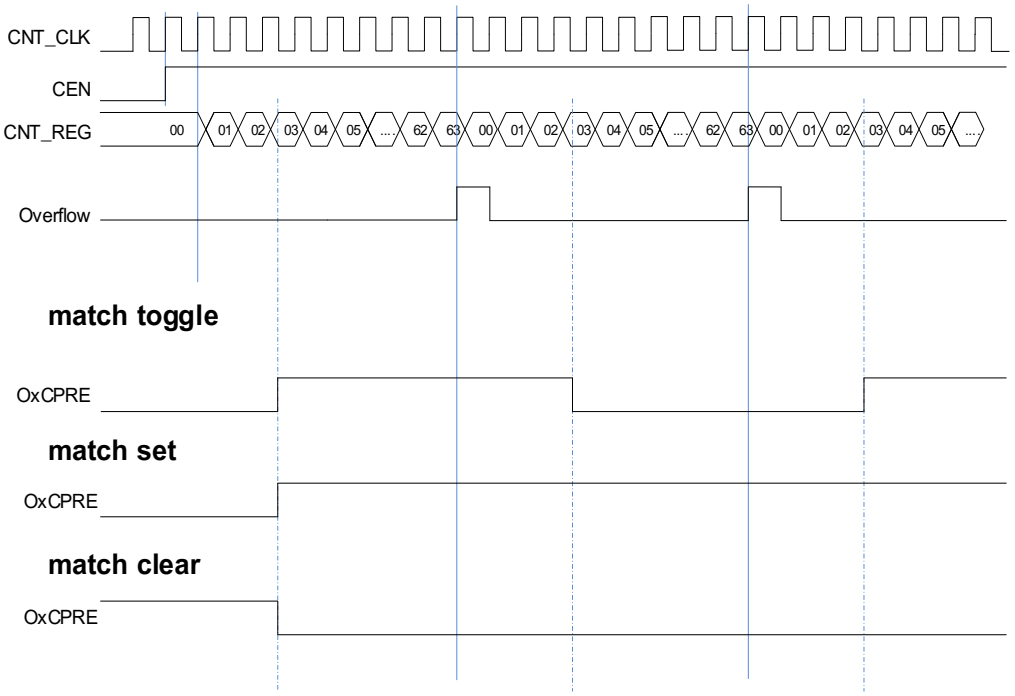
Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV
About the CHxVAL; you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3.

Figure 22-13. Output-compare under three modes



Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can generate PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

Based on the counter mode, we can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

The EAPWM period is determined by TIMERx_CAR and duty cycle is determined by TIMERx_CHxCV. ***Figure 22-14. EAPWM timechart*** shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2*TIMERx_CAR, and duty cycle is by 2*TIMERx_CHxCV. ***Figure 22-15. CAPWM timechart*** shows the CAPWM output and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

Figure 22-14. EAPWM timechart

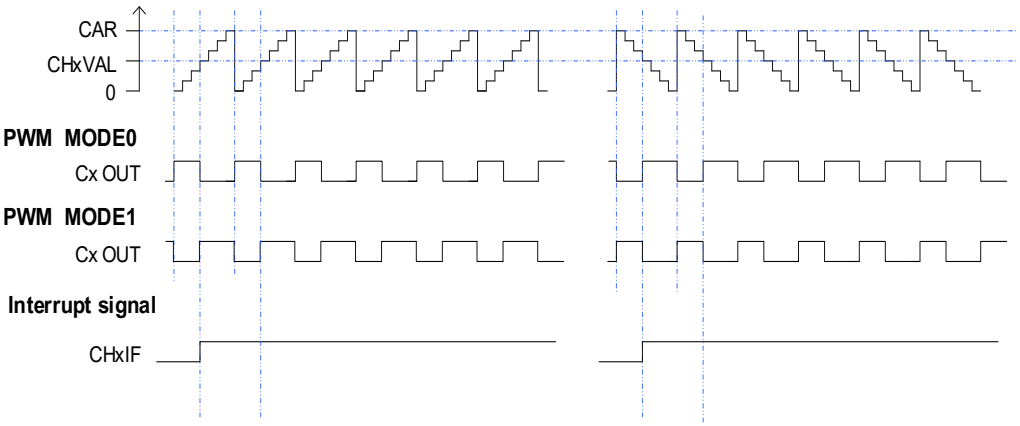
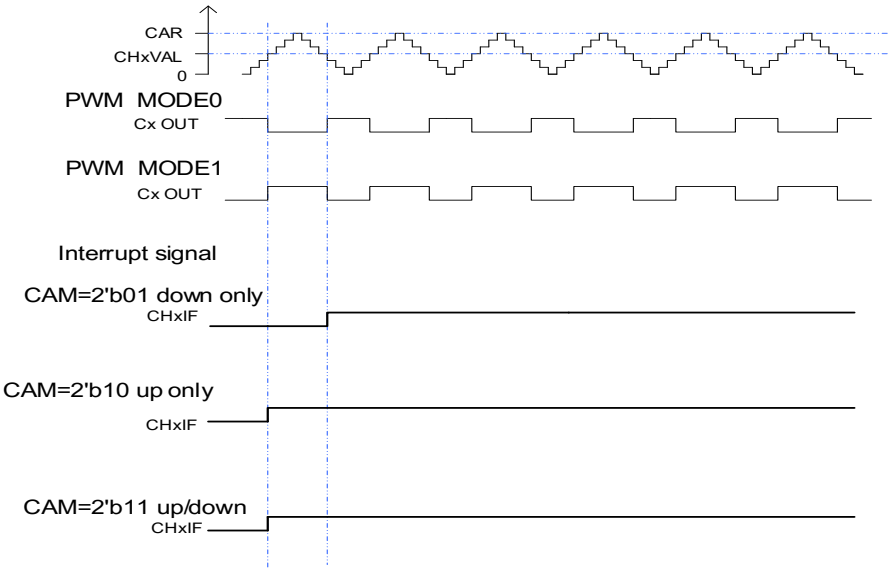


Figure 22-15. CAPWM timechart



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..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 22-2.The Composite PWM pulse width

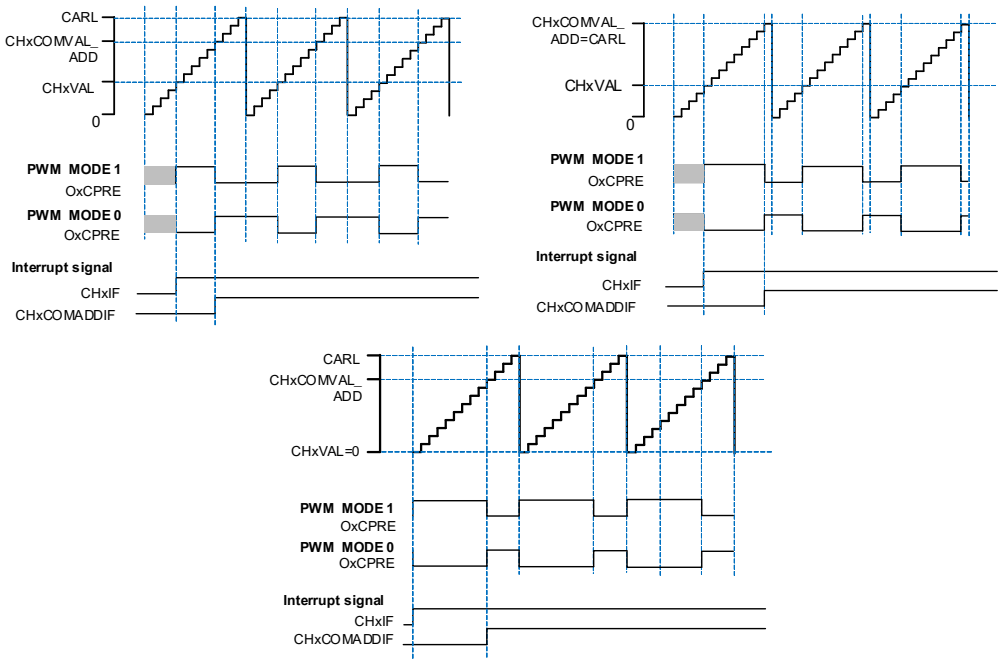
Condition	Mode	PWM pulse width
CHxVAL < CHxCOMVAL_ADD <= CARL	PWM mode 0	(CARL + 0x0001) + (CHxVAL - CHxCOMVAL_ADD)
	PWM mode 1	(CHxCOMVAL_ADD - CHxVAL)
CHxCOMVAL_ADD < CHxVAL <= CARL	PWM mode 0	(CHxVAL - CHxCOMVAL_ADD)
	PWM mode 1	(CARL + 0x0001) + (CHxCOMVAL_ADD - CHxVAL)
(CHxVAL = CHxCOMVAL_ADD <= 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:

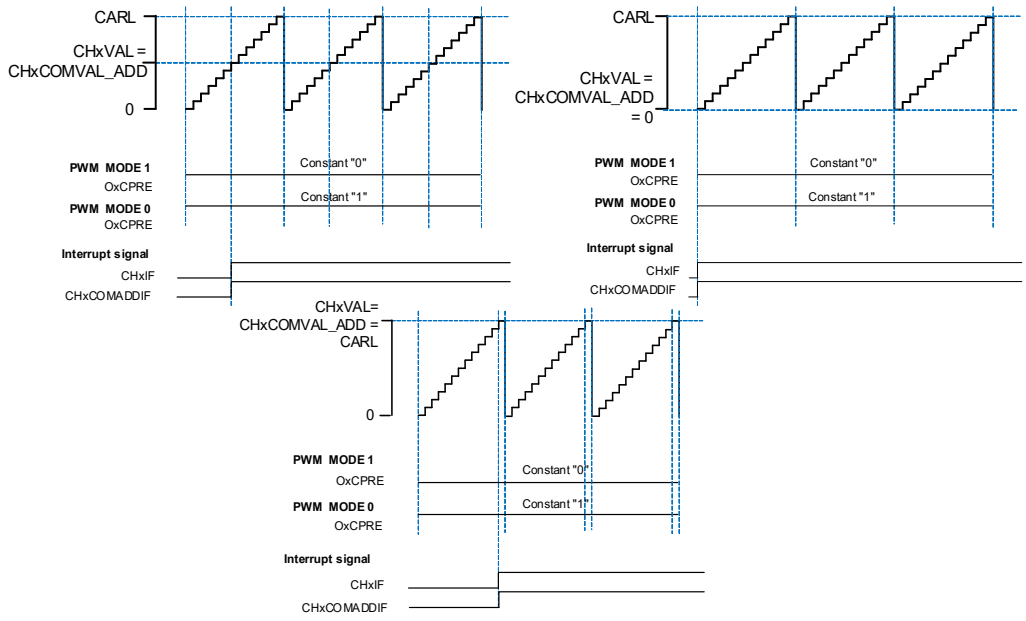
1) CHxVAL < CHxCOMVAL_ADD, and the values of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

Figure 22-16. Channel x output PWM with (CHxVAL < CHxCOMVAL_ADD)



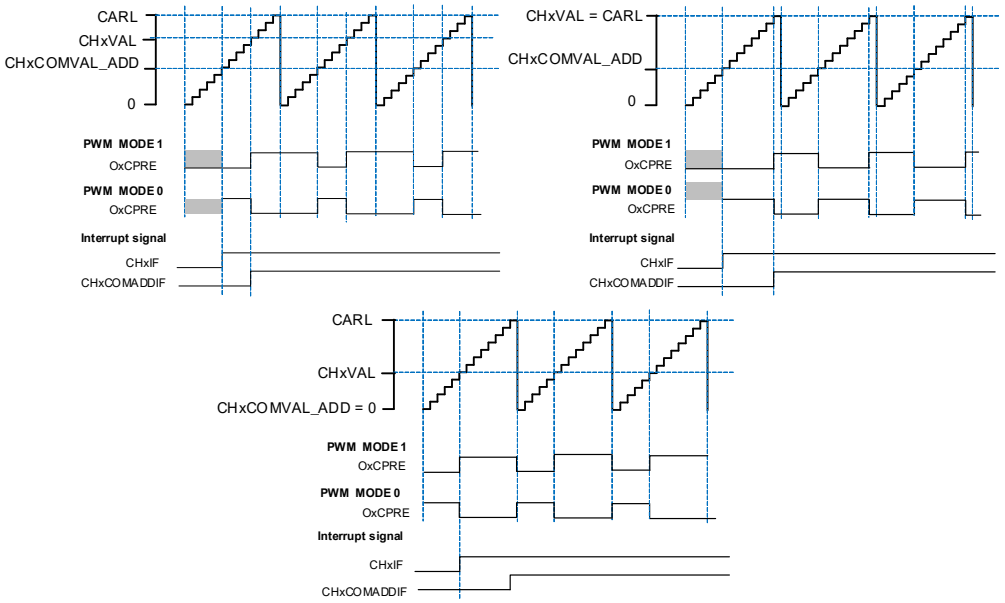
2) CHxVAL = CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

Figure 22-17. Channel x output PWM with (CHxVAL = CHxCOMVAL_ADD)



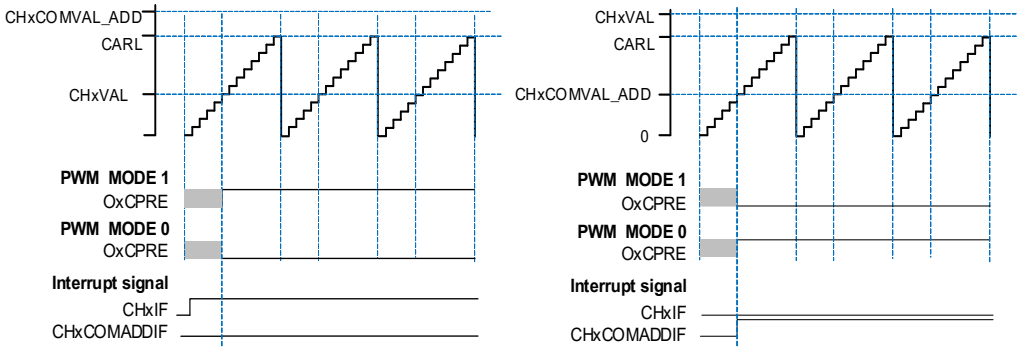
3) CHxVAL > CHxCOMVAL_ADD, and the value of CHxVAL and CHxCOMVAL_ADD between 0 and CARL.

Figure 22-18. Channel x output PWM with (CHxVAL > CHxCOMVAL_ADD)



4) One of the value of CHxVAL and CHxCOMVAL_ADD exceeds CARL.

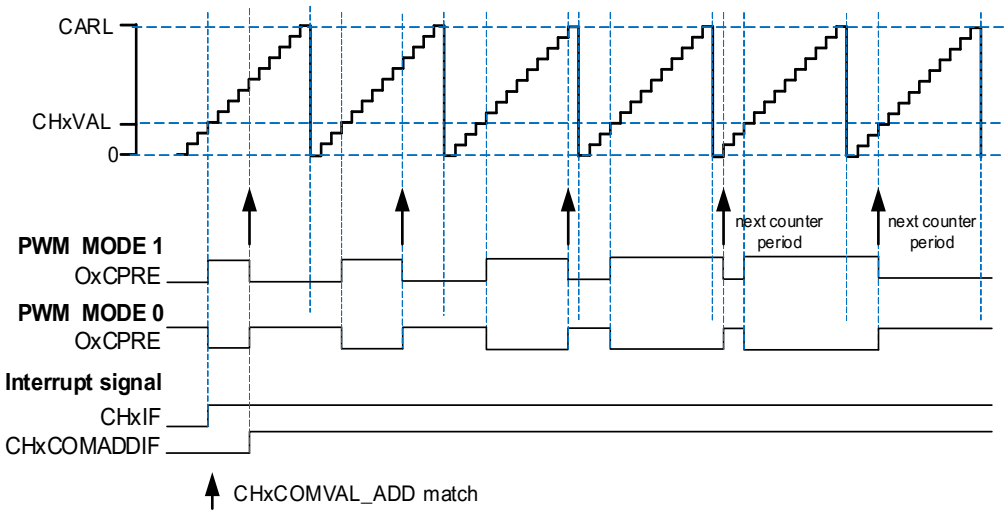
Figure 22-19. Channel x output PWM with CHxVAL or CHxCOMVAL_ADD exceeds CARL



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 22-20. Channel x output PWM duty cycle changing with CHxCOMVAL_ADD* shows the the PWM output and interrupts waveform.

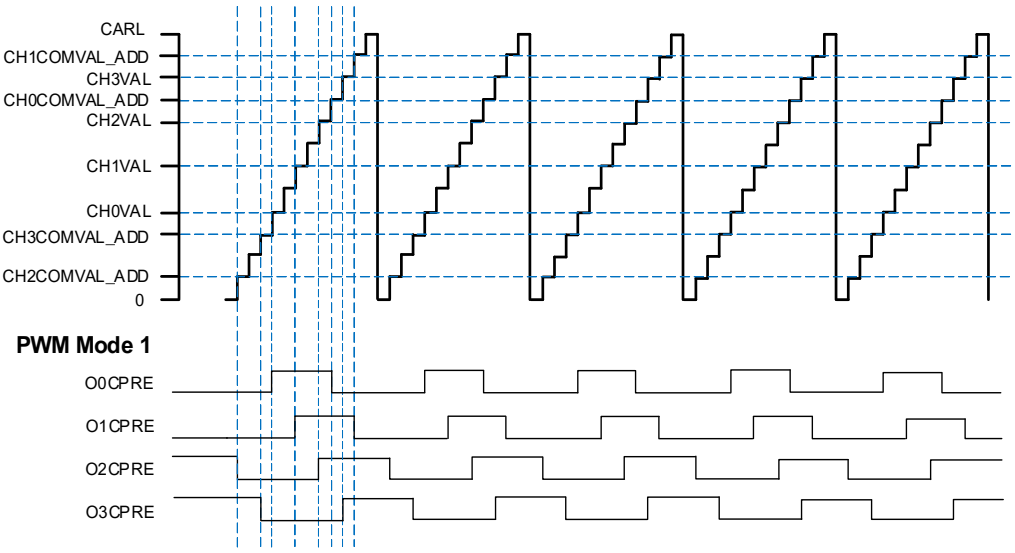
In some cases, the CHxCOMVAL_ADD match can happen on the next counter period (the value of CHxCOMVAL_ADD was written after the counter reaches the value of CHxVAL, and the value of CHxCOMVAL_ADD was less than or equal to the CHxVAL).

Figure 22-20. 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 22-21. Four Channels outputs in Composite PWM mode



Channel output prepare signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting 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. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

Channel output complementary PWM

Function of complementary is for a pair of CHx_O and CHx_ON. Those two output signals cannot be active at the same time. The complementary signals CHx_O and CHx_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx_CHCTL2 register and the POEN, ROS, IOS, ISOx and ISOxN bits in the TIMERx_CCHP and TIMERx_CTL1 registers. The outputs polarity is determined by CHxP and CHxNP bits in the TIMERx_CHCTL2 register.

Table 22-3. Complementary outputs controlled by parameters

Complementary Parameters					Output Status	
POEN	ROS	IOS	CHxEN	CHxNEN	CHx_O	CHx_ON
0	0/1	0	0	0	CHx_O / CHx_ON = LOW CHx_O / CHx_ON output disable(1).	
				1	CHx_O/ CHx_ON output "off-state" (2): 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": 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/CHx_ON = LOW CHx_O/CHx_ON output disable.	
				1	CHx_O = LOW CHx_O output disable.	CHx_ON =OxCPRE⊕(4)CHxNP CHx_ON output enable.
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable.	CHx_ON = LOW CHx_ON output disable.
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable.	CHx_ON =(!OxCPRE)(5)⊕CHxNP. CHx_ON output enable.
	1		0	0	CHx_O = CHxP CHx_O output "off-state".	CHx_ON = CHxNP CHx_ON output "off-state".
				1	CHx_O = CHxP CHx_O output "off-state"	CHx_ON =OxCPRE⊕CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON = CHxNP CHx_ON output "off-state".

				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON =(!OxCPRE)⊕CHxNP CHx_ON 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.

Insertion dead time for complementary PWM

The dead time insertion is enabled when both CHxEN and CHxNEN are 1'b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for all channels. The detail about the delay time, refer to the register TIMERx_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

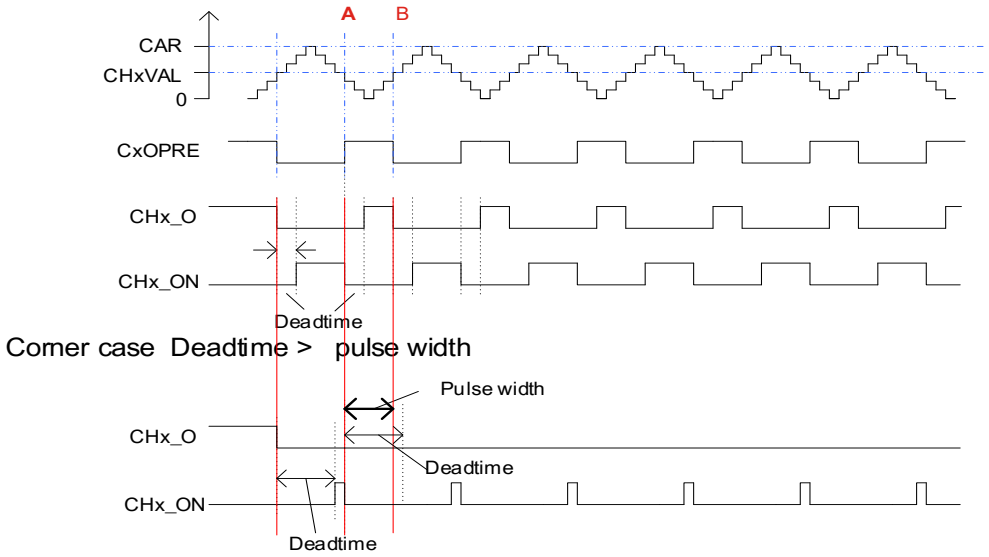
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the ***Figure 22-22. Complementary output with dead-time insertion***. CHx_O signal remains at the low value until the end of the deadtime delay, while CHx_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx_O signal will be cleared at once, while CHx_ON signal remains at the low value 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 CHx_O duty cycle, then the CHx_O signal is always the inactive value. (as show in the ***Figure 22-22. Complementary output with dead-time insertion***.)

- The dead time delay is greater than or equal to the CHx_ON duty cycle, then the CHx_ON signal is always the inactive value.

Figure 22-22. Complementary output with dead-time insertion.



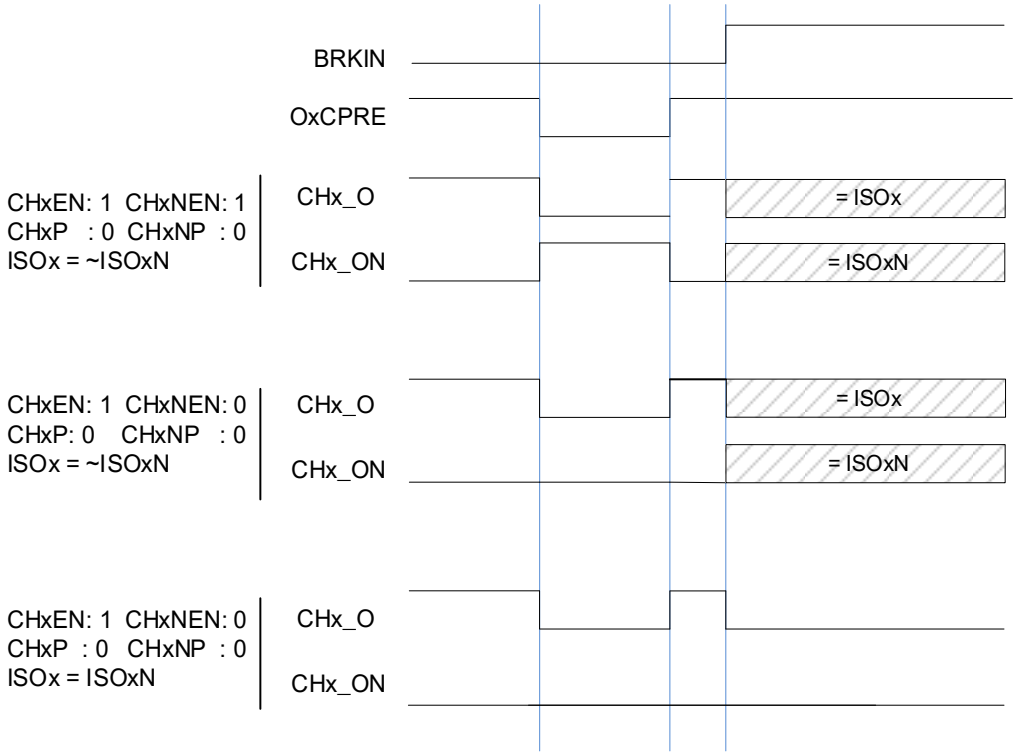
Break mode

In this mode, the output CHx_O and CHx_ON are controlled by the POEN, IOS and ROS bits in the TIMERx_CCHP register, ISOx and ISOxN bits in the TIMERx_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx_O and CHx_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMERx_INTF register is set. If BRKIE is 1, an interrupt generated.

Figure 22-23. Output behavior in response to a break(The break high active)



Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0FE0 and CI1FE1 derived from the TIMERx_CH0 and TIMERx_CH1 pins respectively to interact to control the counter value. The DIR bit is modified during each input source transition. The counter can be changed by the edges of CI0FE0 only, CI1FE1 only or both CI0FE0 and CI1FE1, the selection mode by setting the SMC[2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in ***Table 22-4. Counting direction in different quadrature decoder mode***. The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-period value. Therefore, TIMERx_CAR register must be configured before the counter starts to count.

Table 22-4. Counting direction in different quadrature decoder mode

Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
Quadrature decoder mode 0 SMC[2:0]=3'b000	CI1FE1=1	Down	Up	-	-
	CI1FE1=0	Up	Down	-	-
Quadrature decoder mode 1 SMC [2:0]=3'b010	CI0FE0=1	-	-	Up	Down
	CI0FE0=0	-	-	Down	Up
Quadrature decoder mode 2 SMC [2:0]=3'b011	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 22-24. Counter behavior with CI0FE0 polarity non-inverted in mode 2

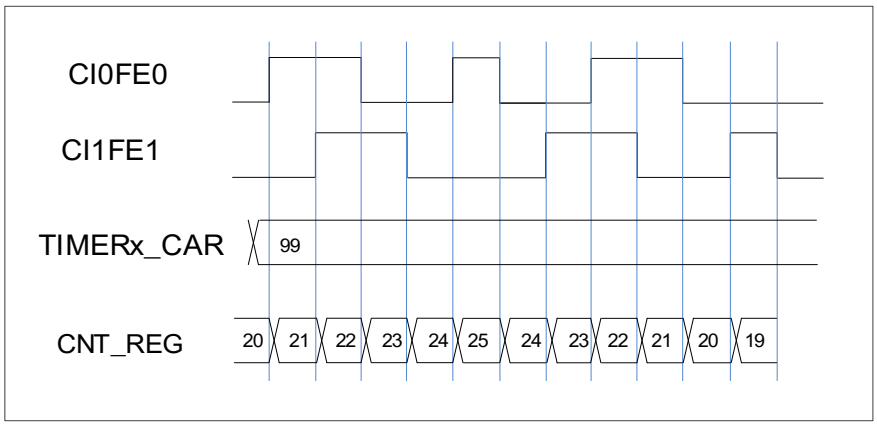
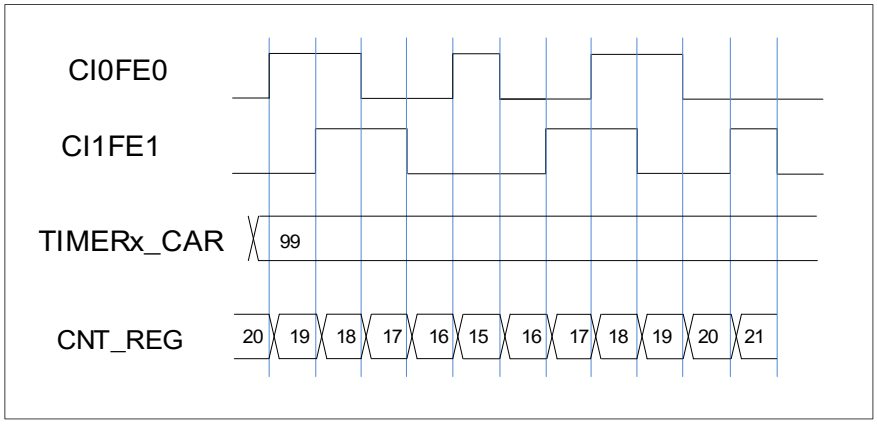


Figure 22-25. Counter behavior with CI0FE0 polarity inverted in mode 2



Hall sensor function

Hall sensor is generally used to control BLDC Motor; advanced timer can support this function.

Figure 22-26. Hall sensor is used to BLDC motor show how to connect. And we can see

we need two timers. First TIMER_in (Advanced/GeneralL0 TIMER) should accept three HALL sensor signals.

Each of the three input of HALL sensors provides a pulse that applied to an input capture pin, can then be analyzed and both speed and position can be deduced.

By the internal connection such as TRGO-ITIx, TIMER_in and TIMER_out can be connected. TIMER_out will generate PWM signal to control BLDC motor's speed based on the ITRx. Then, the feedback circuit is finished, also you change configuration to fit your request.

About the TIMER_in, it need have input XOR function, so you can choose from Advanced/GeneralL0 TIMER.

And TIMER_out need have functions of complementary and Dead-time, so only advanced timer can be chosen. Else, based on the timers' 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)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI0S, then, each of input signal change will make the CI0 toggle. CH0VAL will record the value of counter at that moment.
- Enable ITIx connected to commutation function directly by setting CCUC and CCSE.
- Configuration PWM parameter based on your request.

Figure 22-26. Hall sensor is used to BLDC motor

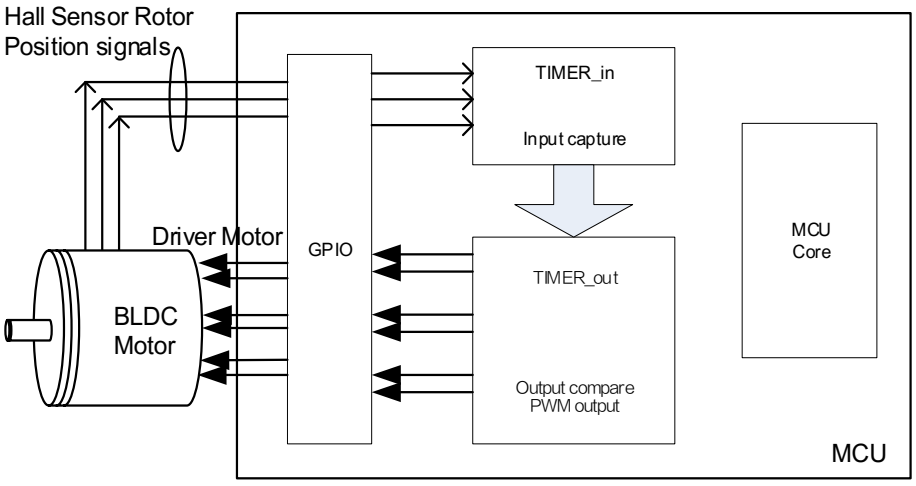
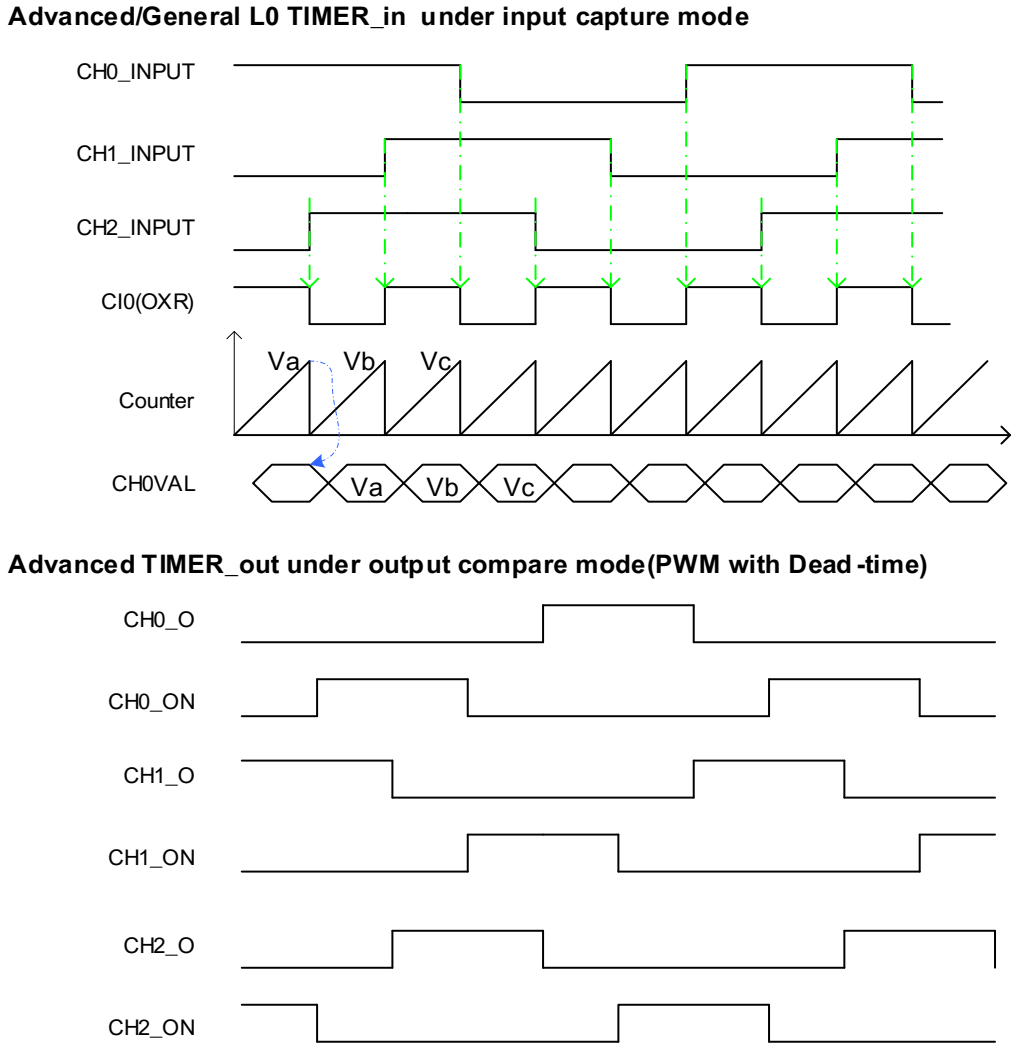


Figure 22-27. Hall sensor timing between two timers

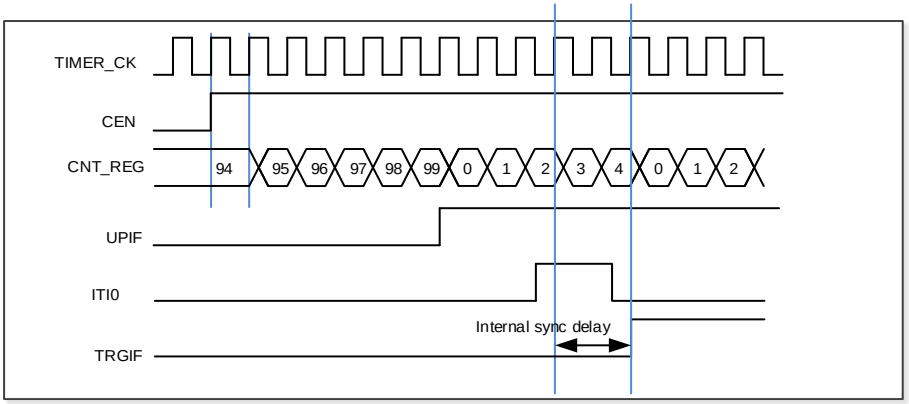
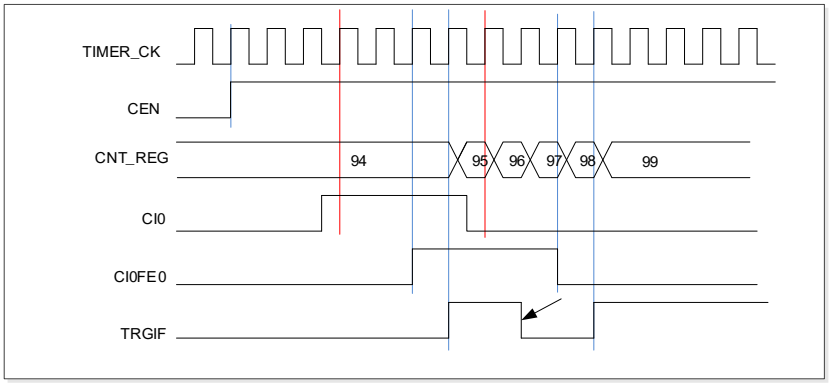


Master-slave management

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

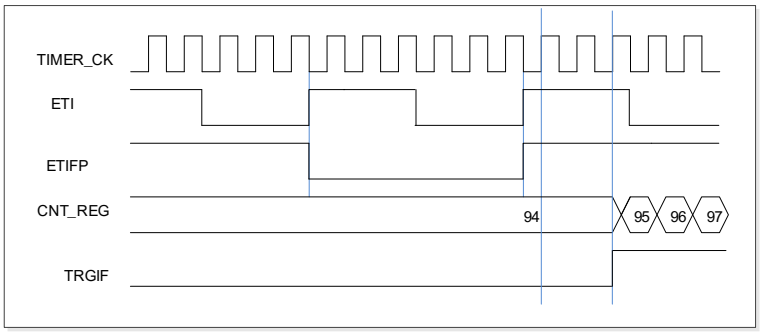
Table 22-5. Slave mode examples

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion. If you choose the ETIF,	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, 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>mode)</td><td>101: CI0FE0
110: CI1FE1
111: ETIFP</td><td>configure the ETP for polarity selection and inversion.</td><td>Filter by ETFC and Prescaler by ETPSC.</td></tr>
<tr><td rowspan="2">Exam1</td><td>Restart mode
The counter can be clear and restart when a rising trigger input.</td><td>TRGS[2:0]=3'b000
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 22-28. Restart mode

</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode
The counter can be paused when the trigger input is low.</td><td>TRGS[2:0]=3'b101
CI0FE0 is the selection.</td><td>TI0S=0(Non-xor) [CH0NP==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 22-29. Pause mode

</td></tr>
<tr><td>Exam3</td><td>Event mode
The counter will start to count when a rising trigger input.</td><td>TRGS[2:0]=3'b111
ETIF is the selection.</td><td>ETP = 0 no polarity change.</td><td>ETPSC = 1, divided by 2.
ETFC = 0, no filter</td></tr>
</table>

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler

Figure 22-30. Event mode



Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

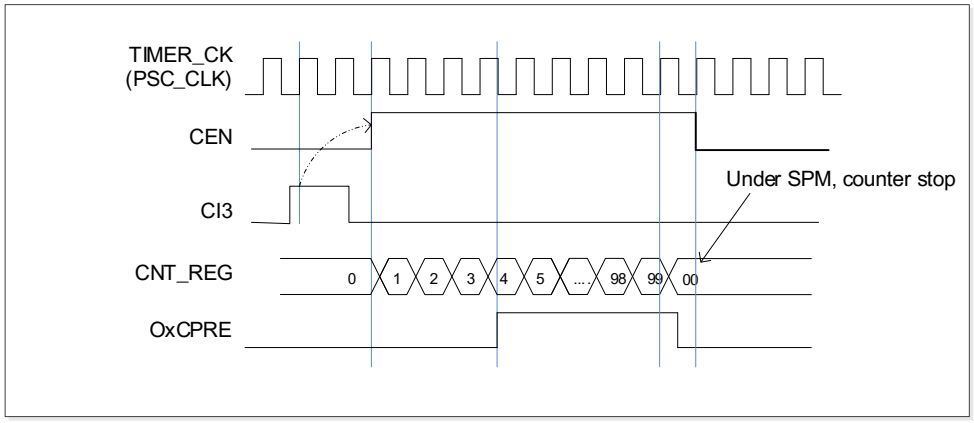
Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals 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 using software, the counter will be stopped and its value held.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0/1 register. 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. The CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

Single pulse mode is also applicable to composite PWM mode (CHxCPWMEN = 1'b1 and CHxMS[2:0] = 3'b000).

Figure 22-31. Single pulse mode, TIMERx_CHxCV = 4, TIMERx_CAR=99 shows an example.

Figure 22-31. Single pulse mode, TIMERx_CHxCV = 4, TIMERx_CAR=99

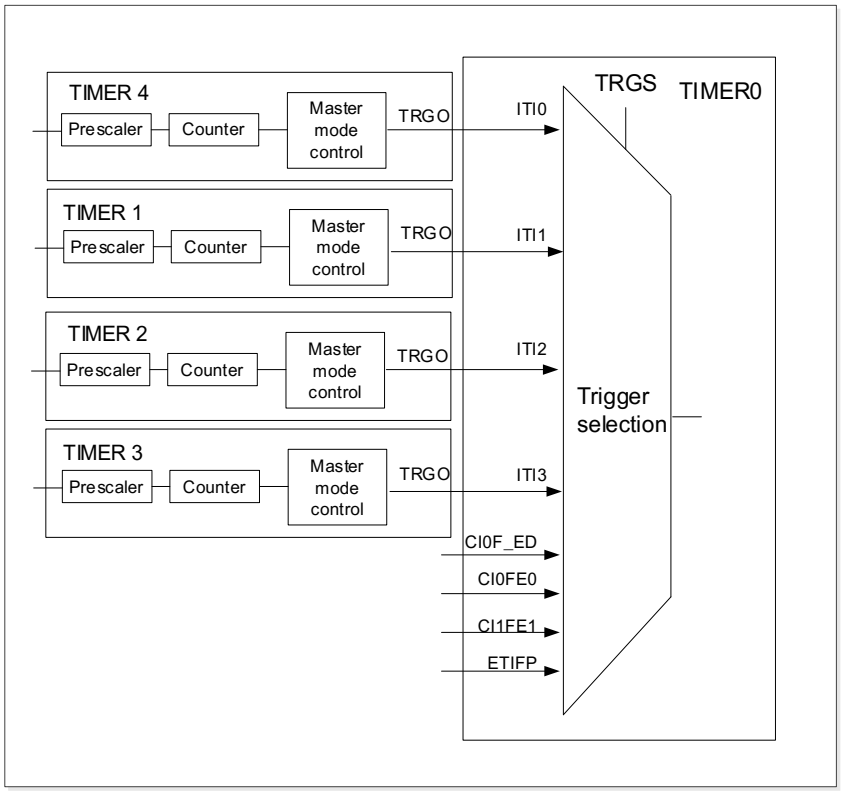


Timers interconnection

Timer can be configured as interconnection, that is, one timer which operate in the master mode outputs TRGO signal to control another timer which operate in the slave mode, TRGO include reset evevt, start evevt, update evevt, capture/compare pulse evevt, compare evevt. slave timer received the ITIx and performs the corresponding mode, include internal clock mode, quadrature decoder mode, restart mode, pause mode, event mode, external clock mode.

Figure 22-32. Timer0 master/slave mode timer example shows the timer0 trigger selection when it is configured in slave mode.

Figure 22-32. Timer0 master/slave mode timer example



Other interconnection examples:

- Timer 2 as prescaler for timer 0

We configure Timer2 as a prescaler for Timer 0. Refer to ***Figure 22-32. Timer0 master/slave mode timer example*** for connections. Do as bellow:

1. Configure Timer2 in master mode and select its update event (UPE) as trigger output (MMC=3'b010 in the TIMER2_CTL1 register). Then timer2 drives a periodic signal on each counter overflow.
2. Configure the Timer2 period (TIMER2_CAR registers).
3. Select the Timer0 input trigger source from Timer2(TRGS=3'b010 in the TIMERx_SMCFG register).
4. Configure Timer0 in external clock mode 0 (SMC=3'b111 in TIMERx_SMCFG register).
5. Start Timer0 by writing '1 in the CEN bit (TIMER0_CTL0 register).
6. Start Timer2 by writing '1 in the CEN bit (TIMER2_CTL0 register).

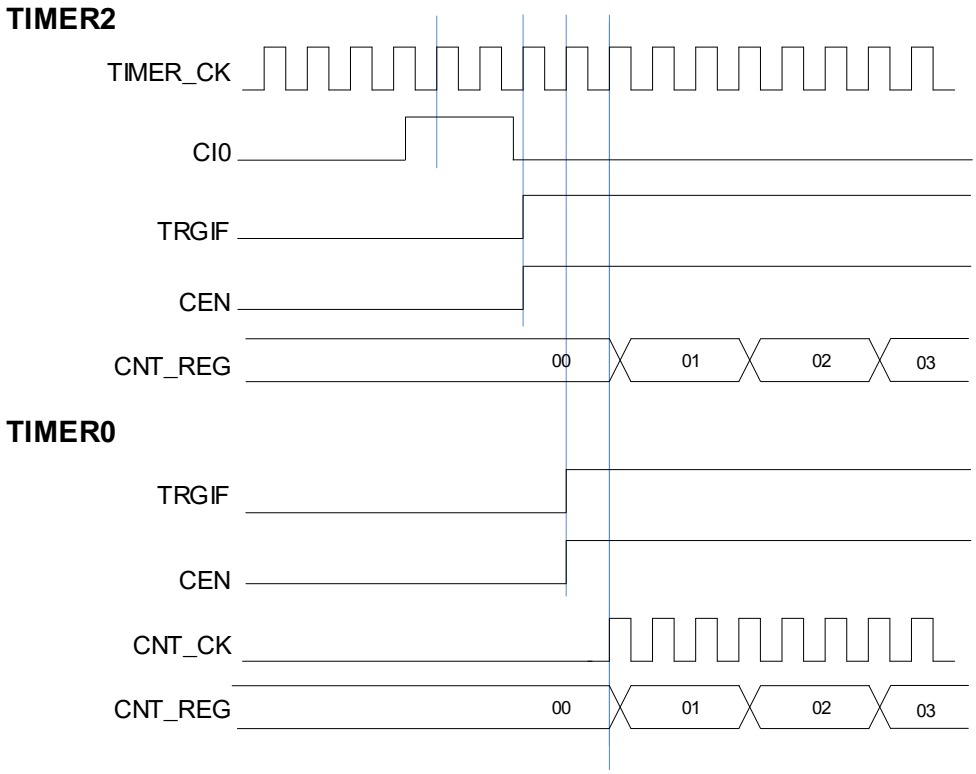
- Using an external trigger to start 2 timers synchronously

We configure the start of Timer0 is triggered by the enable of Timer2, and Timer2 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, Timer2 must be configured in Master/Slave mode. Do as follow:

1. Configure Timer2 slave mode to get the input trigger from CI0 (TRGS=3'b100 in the TIMER2_SMCFG register).
2. Configure Timer2 in event mode (SMC=3'b110 in the TIMER2_SMCFG register).
3. Configure the Timer2 in Master/Slave mode by writing MSM=1 (TIMER2_SMCFG register).
4. Configure Timer0 to get the input trigger from Timer2 (TRGS=3'b010 in the TIMERx_SMCFG register).
5. Configure Timer0 in event mode (SMC=3'b110 in the TIMER0_SMCFG register).

When a rising edge occurs on Timer2's CI0, two timer's counters start counting synchronously on the internal clock and both TRGIF flags are set.

Figure 22-33. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input



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 one 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.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL2 register is set to 1, the TIMERx counter stops.

22.1.4. Register definition

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						CKDIV[1:0]		ARSE	CAM[1:0]		DIR	SPM	UPS	UPDIS	CEN
						rw		rw	rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK_TIMER}$ 01: $f_{DTS}= f_{CK_TIMER}$ /2 10: $f_{DTS}= f_{CK_TIMER}$ /4 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	Counter aligns mode selection 00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit. 01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting down, CHxF bit can be set. 10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting up, CHxF bit can be set. 11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Both when counting up and counting down, CHxF bit can be set.

		After the counter is enabled, cannot be switched from 0x00 to non 0x00.
4	DIR	Direction 0: Count up 1: Count down If the timer work in center-aligned mode or decoder mode, this bit is read only.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
0	CEN	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and decoder mode.

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
ISO3N	ISO3	ISO2N	ISO2	ISO1N	ISO1	ISO0N	ISO0	TI0S	MMC[2:0]			DMAS	CCUC	Reserved	CCSE
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	ISO3N	Idle state of channel 3 complementary output Refer to ISO0N bit
14	ISO3	Idle state of channel 3 output Refer to ISO0 bit
13	ISO2N	Idle state of channel 2 complementary output Refer to ISO0N bit
12	ISO2	Idle state of channel 2 output Refer to ISO0 bit
11	ISO1N	Idle state of channel 1 complementary output Refer to ISO0N bit
10	ISO1	Idle state of channel 1 output Refer to ISO0 bit
9	ISO0N	Idle state of channel 0 complementary output 0: When POEN bit is reset, CH0_ON is set low. 1: When POEN bit is reset, CH0_ON is set high 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 0: When POEN bit is reset, CH0_O is set low. 1: When POEN bit is reset, CH0_O is set high The CH0_O output changes after a dead-time if CH0_ON 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 0: The TIMERx_CH0 pin input is selected as channel 0 trigger input. 1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The

		counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit. 011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output. 100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE. 101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE. 110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE. 111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE.
3	DMAS	DMA request source selection 0: When capture or compare event occurs, the DMA request of channel x is sent 1: When update event occurs, the DMA request of channel x is sent.
2	CCUC	Commutation control shadow register update control When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below: 0: The shadow registers update by when CMTG bit is set. 1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs. When a channel does not have a complementary output, this bit has no effect.
1	Reserved	Must be kept at reset value.
0	CCSE	Commutation control shadow enable 0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled. 1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled. After these bits have been written, they are updated based when commutation event coming. 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).

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
ETP	SMC1	ETPSC[1:0]		ETFC[3:0]				MSM	TRGS[2:0]			Reserved	SMC[2:0]		
rw	rw	rw		rw				rw	rw			rw	rw		

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	ETP	External trigger polarity This bit specifies the polarity of ETI signal 0: ETI is active at rising edge or high level . 1: ETI is active at falling edge or low level .
14	SMC1	Part of SMC for enable External clock mode1. In external clock mode 1, the counter is clocked by any active edge on the ETIFP signal. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case. The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time. **Note:** External clock mode 0 enable is in this register's SMC[2:0] bit-filed.
13:12	ETPSC[1:0]	The prescaler of external trigger The frequency of external trigger signal ETIFP must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETIFP frequency. 00: Prescaler disable. 01: The prescaler is 2. 10: The prescaler is 4. 11: The prescaler is 8.
11:8	ETFC[3:0]	External trigger filter control The external trigger can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the external trigger signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level. The filtering capability configuration is as follows:

EXTFC[3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	

4'b0100	6	$f_{DTS_CK}/2$
4'b0101	8	
4'b0110	6	$f_{DTS_CK}/4$
4'b0111	8	
4'b1000	6	$f_{DTS_CK}/8$
4'b1001	8	
4'b1010	5	$f_{DTS_CK}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS_CK}/32$
4'b1110	6	
4'b1111	8	

7	MSM	Master-slave mode This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together. 0: Master-slave mode disable 1: Master-slave mode enable
6:4	TRGS[2:0]	Trigger selection This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter. 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	Slave mode control 000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high. 001: Quadrature decoder mode 0.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level. 010: Quadrature decoder mode 1.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level. 011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 edge, while the direction depends on each other. 100: Restart mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input.

101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter clock when it is low.
110: Event mode. A rising edge of the trigger input enables the counter.
111: External clock mode 0. The counter counts on the rising edges of the selected trigger.

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											
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 0: Disabled 1: Enabled **Note:** This bit just used in composite PWM mode.
30	CH2COMADDIE	Channel 2 additional compare interrupt enable 0: Disabled 1: Enabled **Note:** This bit just used in composite PWM mode.
29	CH1COMADDIE	Channel 1 additional compare interrupt enable 0: Disabled 1: Enabled **Note:** This bit just used in composite PWM mode.
28	CH0COMADDIE	Channel 0 additional compare interrupt enable 0: Disabled 1: Enabled **Note:** This bit just used in composite PWM mode.
27:15	Reserved	Must be kept at reset value.
14	TRGDEN	Trigger DMA request enable 0: Disabled 1: Enabled
13	CMTDEN	Commutation DMA request enable

		0: Disabled 1: Enabled
12	CH3DEN	Channel 3 capture/compare DMA request enable 0: Disabled 1: Enabled
11	CH2DEN	Channel 2 capture/compare DMA request enable 0: Disabled 1: Enabled
10	CH1DEN	Channel 1 capture/compare DMA request enable 0: Disabled 1: Enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: Disabled 1: Enabled
8	UPDEN	Update DMA request enable 0: Disabled 1: Enabled
7	BRKIE	Break interrupt enable 0: Disabled 1: Enabled
6	TRGIE	Trigger interrupt enable 0: Disabled 1: Enabled
5	CMTIE	commutation interrupt enable 0: Disabled 1: Enabled
4	CH3IE	Channel 3 capture/compare interrupt enable 0: Disabled 1: Enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: Disabled 1: Enabled
2	CH1IE	Channel 1 capture/compare interrupt enable 0: Disabled 1: Enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: Disabled 1: Enabled

0	UPIE	Update interrupt enable 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).



Bits	Fields	Descriptions
31	CH3COMADDIF	Channel 3 additional compare interrupt flag. Refer to CH0COMADDIF description.
30	CH2COMADDIF	Channel 2 additional compare interrupt flag. Refer to CH0COMADDIF description.
29	CH1COMADDIF	Channel 1 additional compare interrupt flag. Refer to CH0COMADDIF description.
28	CH0COMADDIF	Channel 0 additional compare interrupt flag. This flag is set by hardware and cleared by software. If channel 0 is in output mode, this flag is set when a compare event occurs. 0: No channel 0 output compare interrupt occurred 1: Channel 0 output compare interrupt occurred **Note:** This flag just used in composite PWM mode.
27:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag 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. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8	Reserved	Must be kept at reset value.
7	BRKIF	Break interrupt flag When the break input is inactive, the bit is set by hardware. When the break input is inactive, the bit can be cleared by software. 0: No active level break has been detected. 1: An active level has been detected.
6	TRGIF	Trigger interrupt flag This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5	CMTIF	Channel commutation interrupt flag This flag is set by hardware when channel's commutation event occurs, and cleared by software 0: No channel commutation interrupt occurred 1: Channel commutation interrupt occurred
4	CH3IF	Channel 3 's capture/compare interrupt flag Refer to CH0IF description
3	CH2IF	Channel 2 's capture/compare interrupt flag Refer to CH0IF description
2	CH1IF	Channel 1 's capture/compare interrupt flag Refer to CH0IF description
1	CH0IF	Channel 0 's capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 0: No Channel 0 interrupt occurred 1: Channel 0 interrupt occurred
0	UPIF	Update interrupt flag This bit is set by hardware on an update event 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
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								BRKG	TRGG	CMTG	CH3G	CH2G	CH1G	CH0G	UPG
								w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31	CH3COMADDG	Channel 3 additional compare event generation. Refer to CH0COMADDG description.
30	CH2COMADDG	Channel 2 additional compare event generation. Refer to CH0COMADDG description.
29	CH1COMADDG	Channel 1 additional compare event generation. Refer to CH0COMADDG description.
28	CH0COMADDG	Channel 0 additional compare event generation. This bit is set by software to generate an additional compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0COMADDIF flag will be set, and the corresponding interrupt will be sent if enabled. 0: No generate a channel 0 additional compare event 1: Generate a channel 0 additional compare event **Note:** This bit just used in composite PWM mode.
27:8	Reserved	Must be kept at reset value.
7	BRKG	Break event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a break event 1: Generate a break event
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event

		1: Generate a trigger event
5	CMTG	Channel commutation event generation This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1). 0: No affect 1: Generate channel's c/c control update event
4	CH3G	Channel 3's capture or compare event generation Refer to CH0G description
3	CH2G	Channel 2's capture or compare event generation Refer to CH0G description
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	Update event generation This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time. 0: No generate an update event 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
Reserved		CH1COM ADDSEN	CH0COM ADDSEN	Reserved											
Reserved															
		rw	rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COMCEN	CH1COMCTL[2:0]			CH1COMSEN	CH1COMFEN	CH1MS[1:0]		CH0COMCEN	CH0COMCTL[2:0]			CH0COMSEN	CH0COMFEN	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
31:30	Reserved	Must be kept at reset value.
29	CH1COMADDSEN	Channel 1 additional compare output shadow enable Refer to CH0COMADDSEN description.
28	CH0COMADDSEN	Channel 0 additional compare output shadow enable 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 1: Channel 0 additional compare output shadow enabled The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 000.
27:16	Reserved	Must be kept at reset value.
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMFEN description
9:8	CH1MS[1:0]	Channel 1 mode selection 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. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. **Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable.

		When this bit is set, if the ETIFP signal is detected as high level, the O0CPRE signal will be cleared. 0: Channel 0 output compare clear disable 1: Channel 0 output compare clear enable
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. **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 is adjusted from "Timing" mode to "PWM" mode or the comparison result changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).
3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1) This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH0COMFEN	Channel 0 output compare fast enable

0: Channel 0 output quickly compare disable.Only the valid edge of the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the result of the comparison to activate CH0_O output is 5 clock cycles.
1: Channel 0 output quickly compare enable.Both the valid edge of the trigger input signal and the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.

1:0	CH0MS[1:0]	Channel 0 I/O mode selection 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 0 is programmed as output mode 01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0 10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0 11: Channel 0 is programmed as input mode, IS0 is connected to ITS **Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture filter control The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH0CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$

4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	8	

3:2 CH0CAPPSC[1:0] Channel 0 input capture prescaler

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 clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH0MS[1:0] Channel 0 mode selection

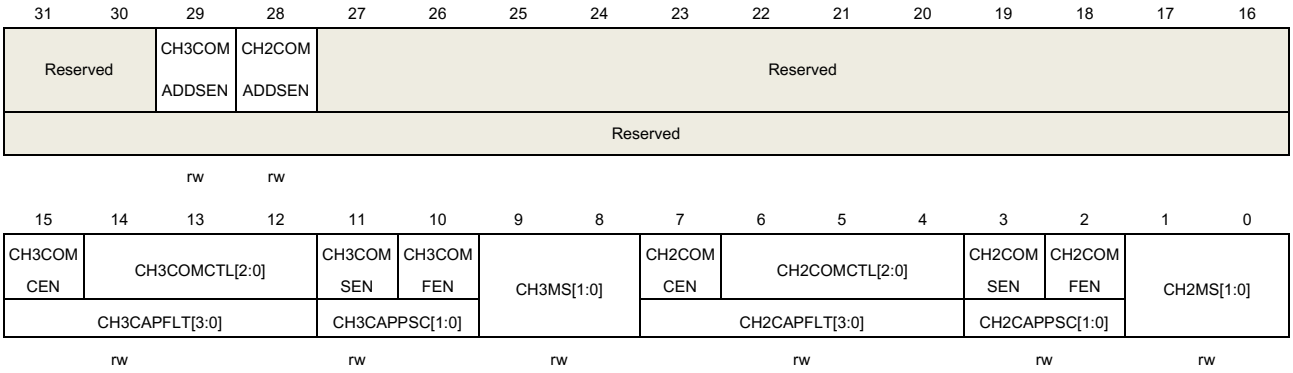
Same as Output compare mode

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
31:30	Reserved	Must be kept at reset value.
29	CH3COMADDSEN	Channel 3 additional compare output shadow enable Refer to CH2COMADDSEN description.
28	CH2COMADDSEN	Channel 2 additional compare output shadow enable

		When this bit is set, the shadow register of TIMERx_CH2COMV_ADD register which updates at each update event will be enabled. 0: Channel 2 additional compare shadow disabled 1: Channel 2 additional compare shadow enabled The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set). This bit cannot be modified when PROT[1:0] bit-field in TIMERx_CCHP register is 11 and CH2MS bit-field is 000.
27:16	Reserved	Must be kept at reset value.
15	CH3COMCEN	Channel 3 output compare clear enable Refer to CH0COMCEN description
14:12	CH3COMCTL[2:0]	Channel 3 compare output control Refer to CH0COMCTL description
11	CH3COMSEN	Channel 3 output compare shadow enable Refer to CH0COMSEN description
10	CH3COMFEN	Channel 3 output compare fast enable Refer to CH0COMFEN description
9:8	CH3MS[1:0]	Channel 3 mode selection 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. (CH3EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 3 is programmed as output mode 01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3 10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3 11: Channel 3 is programmed as input mode, IS3 is connected to ITS. **Note:** When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, if the ETIFP signal is detected as high level, the O2CPRE signal will be cleared. 0: Channel 2 output compare clear disable 1: Channel 2 output compare clear enable
6:4	CH2COMCTL[2:0]	Channel 2 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits. 000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT. 001: Set the channel output. O2CPRE signal is forced high when the counter is

		equals to the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV. 100: Force low. O2CPRE is forced to low level. 101: Force high. O2CPRE is forced to high level. 110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low when the counter is larger than TIMERx_CH2CV, and high otherwise. 111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller than TIMERx_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx_CH2CV, and low otherwise. **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 is adjusted from "Timing" mode to "PWM" mode or the comparison result changes. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH2MS bit-filed is 00(COMPARE MODE).
3	CH2COMSEN	Channel 2 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled. 0: Channel 2 output compare shadow disable 1: Channel 2 output compare shadow enable The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1) This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.
2	CH2COMFEN	Channel 2 output compare fast enable 0: Channel 2 output quickly compare disable.Only the valid edge of the comparison result will generate a comparison match, and CH2_O is set to the compare level.The minimum delay from an edge on the result of the comparison to activate CH2_O output is 5 clock cycles. 1: Channel 2 output quickly compare enable.Both the valid edge of the trigger input signal and the comparison result will generate a comparison match, and CH2_O is set to the compare level.The minimum delay from an edge on the trigger input to activate CH2_O output is 3 clock cycles.
1:0	CH2MS[1:0]	Channel 2 I/O mode selection 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. (CH2EN bit in

TIMERx_CHCTL2 register is reset).).

00: Channel 2 is programmed as output mode

01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2

10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2

11: Channel 2 is programmed as input mode, IS2 is connected to ITS.

Note: When CH2MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:12	CH3CAPFLT[3:0]	Channel 3 input capture filter control Refer to CH0CAPFLT description
11:10	CH3CAPPSC[1:0]	Channel 3 input capture prescaler Refer to CH0CAPPSC description
9:8	CH3MS[1:0]	Channel 3 mode selection Same as Output compare mode
7:4	CH2CAPFLT[3:0]	Channel 2 input capture filter control The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI2 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH2CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	8	

3:2	CH2CAPPSC[1:0]	Channel 2 input capture prescaler 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 clear. 00: Prescaler disable, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on every 8 channel input edges
1:0	CH2MS[1:0]	Channel 2 mode selection Same as Output compare mode

Channel control register 2 (TIMERx_CHCTL2)

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															
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
CH3NP	CH3NEN	CH3P	CH3EN	CH2NP	CH2NEN	CH2P	CH2EN	CH1NP	CH1NEN	CH1P	CH1EN	CH0NP	CH0NEN	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	CH3NP	Channel 3 complementary output polarity Refer to CH0NP description
14	CH3NEN	Channel 3 complementary output enable Refer to CH0NEN description
13	CH3P	Channel 3 capture/compare function polarity Refer to CH0P description
12	CH3EN	Channel 3 capture/compare function enable Refer to CH0EN description
11	CH2NP	Channel 2 complementary output polarity Refer to CH0NP description
10	CH2NEN	Channel 2 complementary output enable Refer to CH0NEN description
9	CH2P	Channel 2 capture/compare function polarity Refer to CH0P description
8	CH2EN	Channel 2 capture/compare function enable

		Refer to CH0EN description
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	CH1NEN	Channel 1 complementary output enable Refer to CH0NEN description
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 0 complementary output high level is active level 1: Channel 0 complementary output low level is active level When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	CH0NEN	Channel 0 complementary output enable When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0. 0: Channel 0 complementary output disabled 1: Channel 0 complementary output enabled
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted. 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 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. 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
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]	This bit-filed indicates the current counter value. Writing to this bit-filed 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															

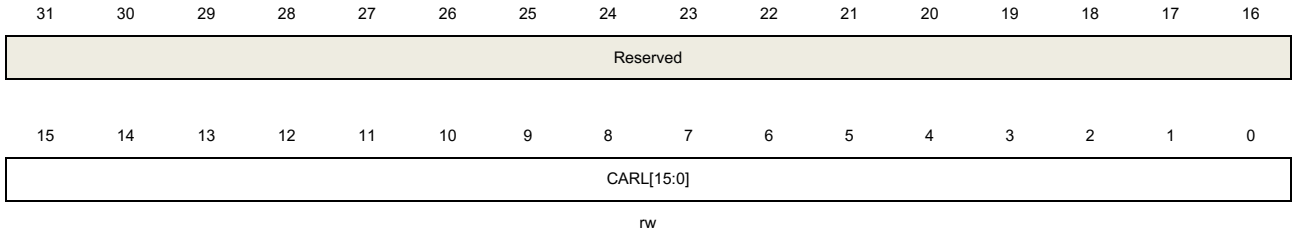
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	PSC[15:0]	Prescaler value of the counter clock The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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



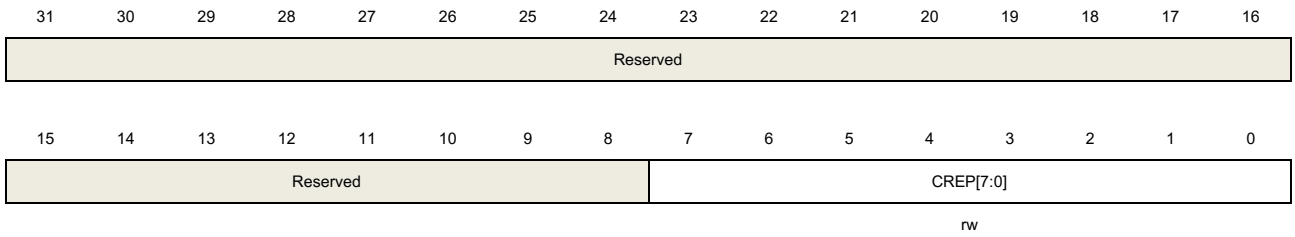
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

Counter repetition register (TIMERx_CREP)

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	CREP[7:0]	Counter repetition value This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled.

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 or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates 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).

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
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH1VAL[15:0]	Capture or compare value of channel1 When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates 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
Reserved															

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	Reserved	Must be kept at reset value.
15:0	CH2VAL[15:0]	Capture or compare value of channel 2 When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates 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
Reserved															

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	Reserved	Must be kept at reset value.
15:0	CH3VAL[15:0]	Capture or compare value of channel 3 When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POEN	OAEN	BRKP	BRKEN	ROS	IOS	PROT[1:0]		DTCFG[7:0]							
rw	rw	rw	rw	rw	rw	rw		rw							

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	POEN	Primary output enable The bit can be set to 1 by: - Write 1 to this bit - If OAEN is set to 1, this bit is set to 1 at the next update event. The bit can be cleared to 0 by: - Write 0 to this bit - Valid fault input (asynchronous). When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set. 0: Disable channel outputs (CHxO or CHxON). 1: Enabled channel outputs (CHxO or CHxON). **Note:** This bit is only valid when CHxMS=2'b00.
14	OAEN	Output automatic enable 0: The POEN bit can only be set by software. 1: POEN can be set at the next update event, if the break input is not active. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
13	BRKP	Break input polarity This bit specifies the polarity of the BRKIN input signal. 0: BRKIN input active low 1; BRKIN input active high
12	BRKEN	Break input enable This bit can be set to enable the BRKIN and CKM clock failure event inputs. 0: Break inputs disabled 1; Break inputs enabled This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
11	ROS	Run mode "off-state" enable 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 22-5***. 0: "off-state" disabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is output disabled.

1: "off-state" enabled. If the CHxEN or CHxNEN bit is reset, the corresponding channel is "off-state".
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
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 22-6.***
0: "off-state" disabled. If the CHxEN/CHxNEN bits are both reset, the channels are output disabled.
1: "off-state" enabled. No matter the CHxEN/CHxNEN bits, the channels are "off-state".
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
This bit-filed specifies the write protection property of registers.
00: protect disable. No write protection.
01: PROT mode 0.The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.
10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected.
11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/ CHxCOMSEN bits in TIMERx_CHCTL0/1 registers (if the related channel is configured in output) are writing protected.
This bit-field can be written only once after the reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected.

7:0 DTCFG[7:0]

Dead time configure
The relationship between DTVAL value and the duration of dead-time is as follow:

DTCFG[7:5]	The duration of dead-time
3'b0xx	DTCFG[7:0] * t_{DTS_CK}
3'b10x	(64+ DTCFG[5:0]) * t_{DTS_CK} *2
3'b110	(32+ DTCFG[4:0]) * t_{DTS_CK} *8
3'b111	(32+ DTCFG[4:0]) * t_{DTS_CK} *16

Note:
1. t_{DTS_CK} is the period of DTS_CK which is configured by CKDIV[1:0] in TIMERx_CTL0.
2. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.

DMA configuration register (TIMERx_DMACFG)

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															

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

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001.
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.

DMA transfer buffer register (TIMERx_DMATB)

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
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	DMATB[15:0]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

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 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. **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 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).

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 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. **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															

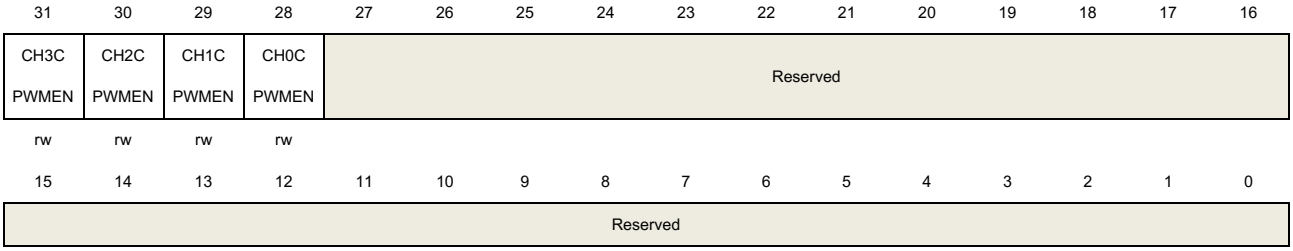
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH3COMVAL_ADD [15:0]	Additional compare value of channel 3 When channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates by every update event. **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).



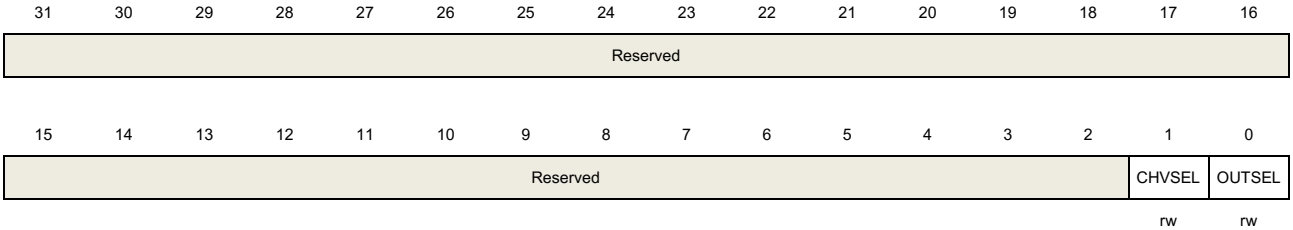
Bits	Fields	Descriptions
31	CH3CPWMEN	Channel 3 composite PWM mode enable 0: Disabled 1: Enabled
30	CH2CPWMEN	Channel 2 composite PWM mode enable 0: Disabled 1: Enabled
29	CH1CPWMEN	Channel 1 composite PWM mode enable 0: Disabled 1: Enabled
28	CH0CPWMEN	Channel 0 composite PWM mode enable 0: Disabled 1: Enabled
27:0	Reserved	Must be kept at reset value.

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 This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	OUTSEL	The output value selection This bit-field set and reset by software 1: If POEN and IOS is 0, the output disabled 0: No effect

22.2. General level0 timer (TIMERx, x=1, 2, 3, 4)

22.2.1. Overview

The general level0 timer module (Timer1, 2, 3, 4) is a four-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level0 time reference is 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 to count or time external events that drive other timers.

Timer and timer are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

22.2.2. Characteristics

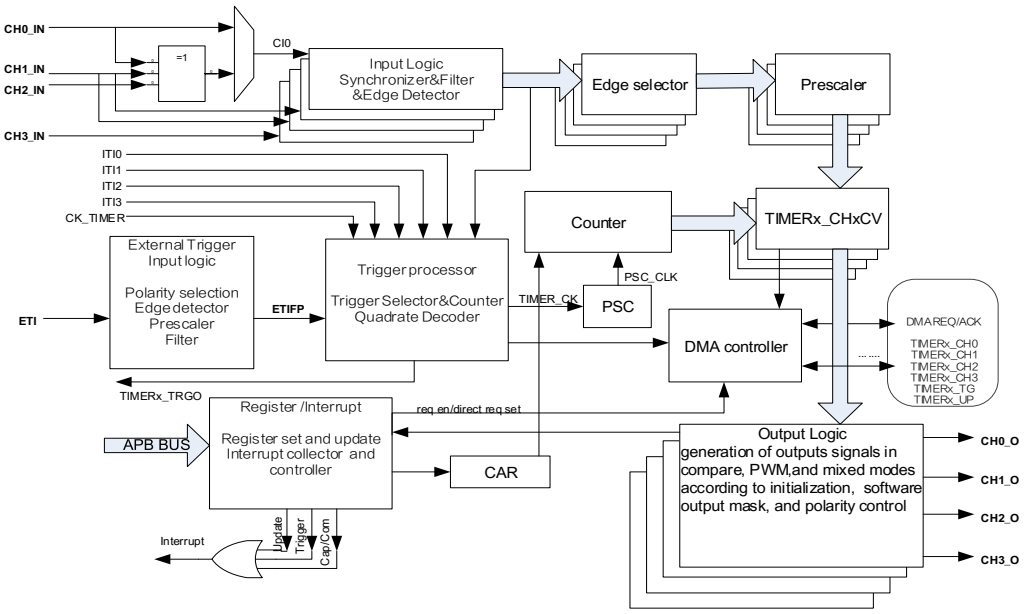
- Total channel num: 4.
- Counter width: 16bit (TIMER2&3), 32bit (TIMER1&4).
- Source of count clock is selectable:
  internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:
  Input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Auto-reload function.
- Interrupt output or DMA request on: update, trigger event, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer master-slave management.

22.2.3. Function overview

Block diagram

Figure 22-34. General Level 0 timer block diagram provides details on the internal configuration of the general level0 timer.

Figure 22-34. General Level 0 timer block diagram



Clock source configuration

The general level0 TIMER has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

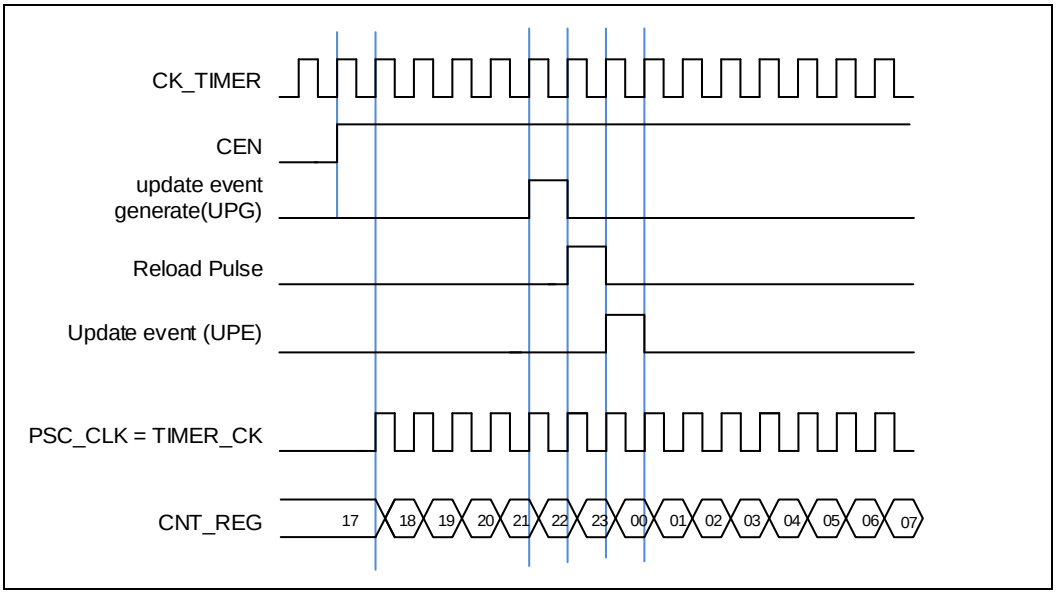
- SMC [2:0] == 3'b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when the SMC [2:0] == 3'b000. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU.

If the SMC [2:0] in the TIMERx_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx_SMCFG register and described as follows. When the SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

Figure 22-35. Timing chart of internal clock divided by 1



- SMC [2:0] == 3'b111(external clock mode 0). External input pin source

The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CI0/TIMERx_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

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 SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

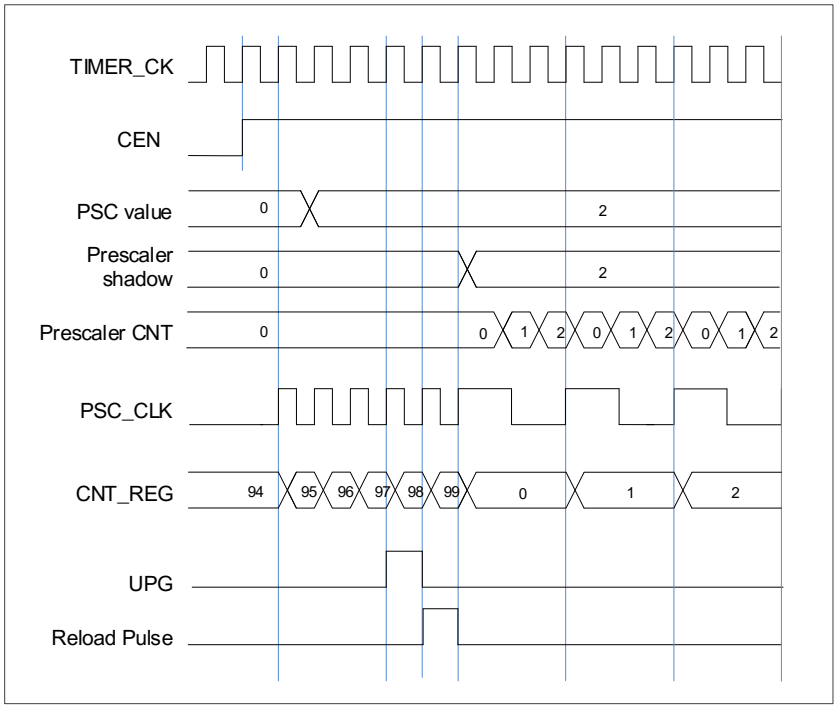
- SMC1== 1'b1(external clock mode 1). External input pin source (ETI)

The TIMER_CK, driven 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 set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the clock source is selected to come from the ETI signal, the trigger controller including the edge detection circuitry will generate a clock pulse during each ETI signal rising edge to clock the counter prescaler.

Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 22-36. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMERx_CAR=0x99.

Figure 22-37. Timing chart of up counting mode, PSC=0/2

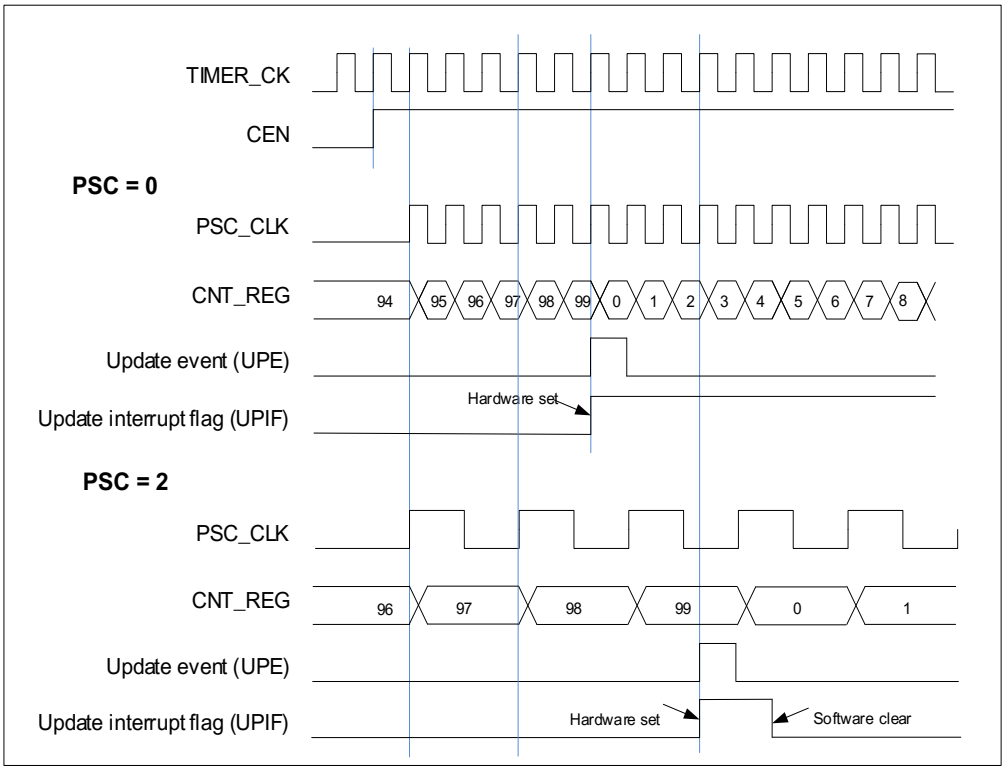
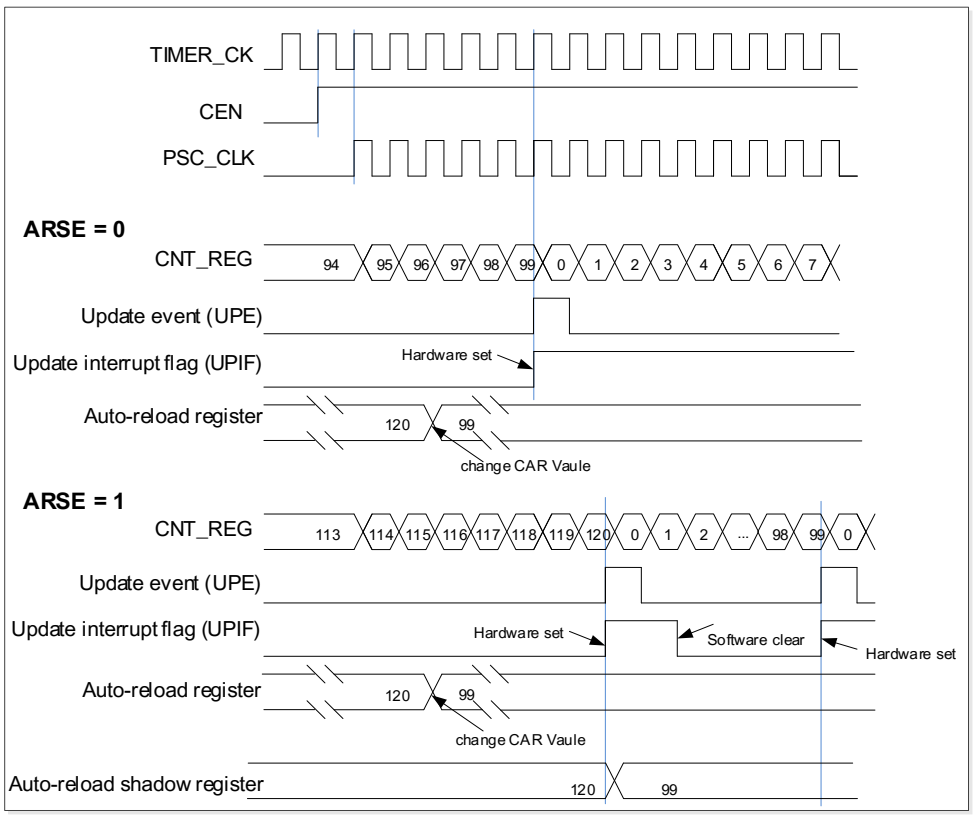


Figure 22-38. Timing chart of up counting mode, change TIMERx_CAR ongoing



Counter down counting

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx_CAR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter will start counting down from the counter-reload value. The update event is generated at each counter underflow. 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 generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMERx_CAR=0x99.

Figure 22-39. Timing chart of down counting mode, PSC=0/2

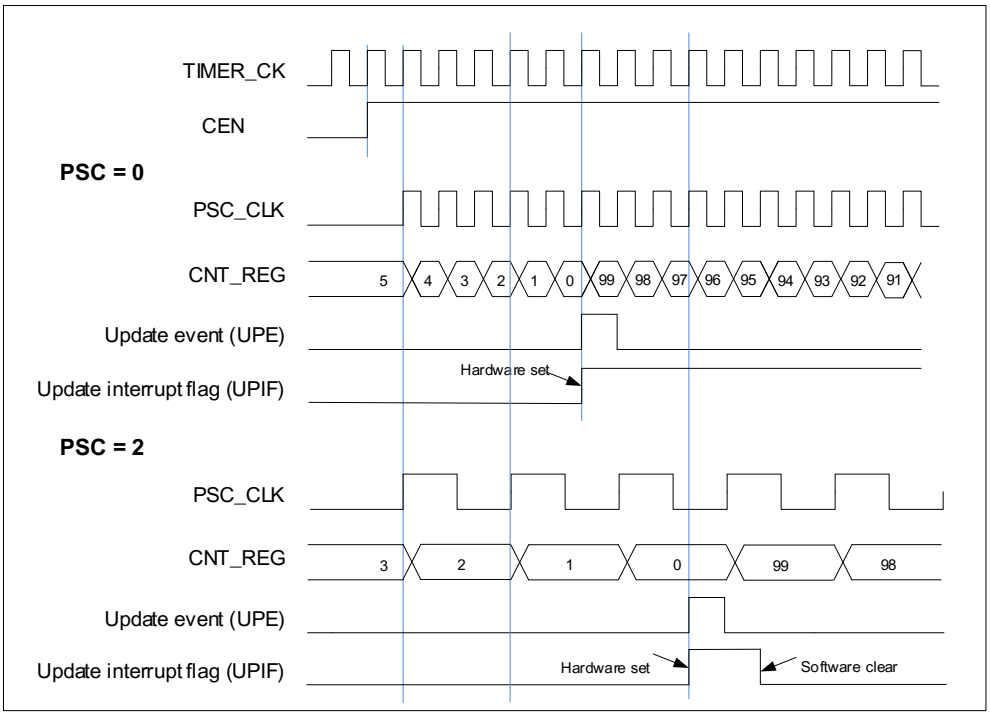
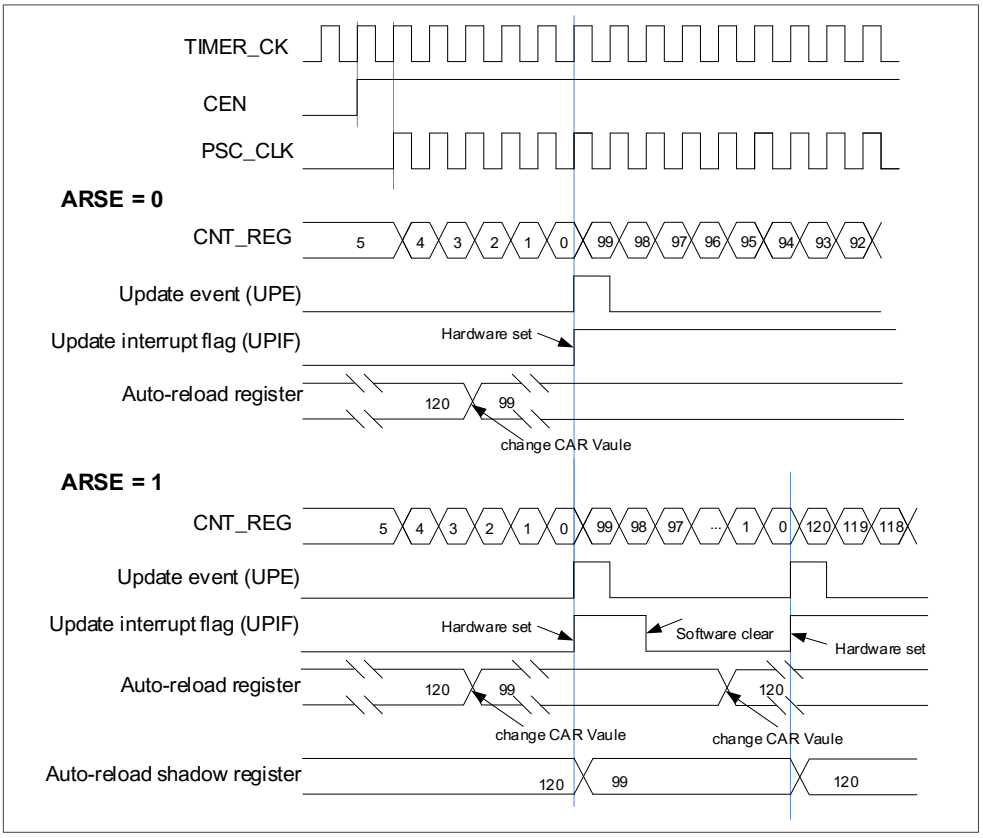


Figure 22-40. Timing chart of down counting mode, change TIMERx_CAR ongoing



Counter center-aligned counting

In this 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 the counter-reload value subtract 1 in the up-counting mode and generates an underflow event when the counter counts to 1 in the down-counting mode. The counting direction bit DIR in the TIMERx_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode.

Setting the UPG bit in the TIMERx_SWEVG register will initialize the counter value to 0 irrespective of whether the counter is counting up or down in the center-align counting mode and generates an update event.

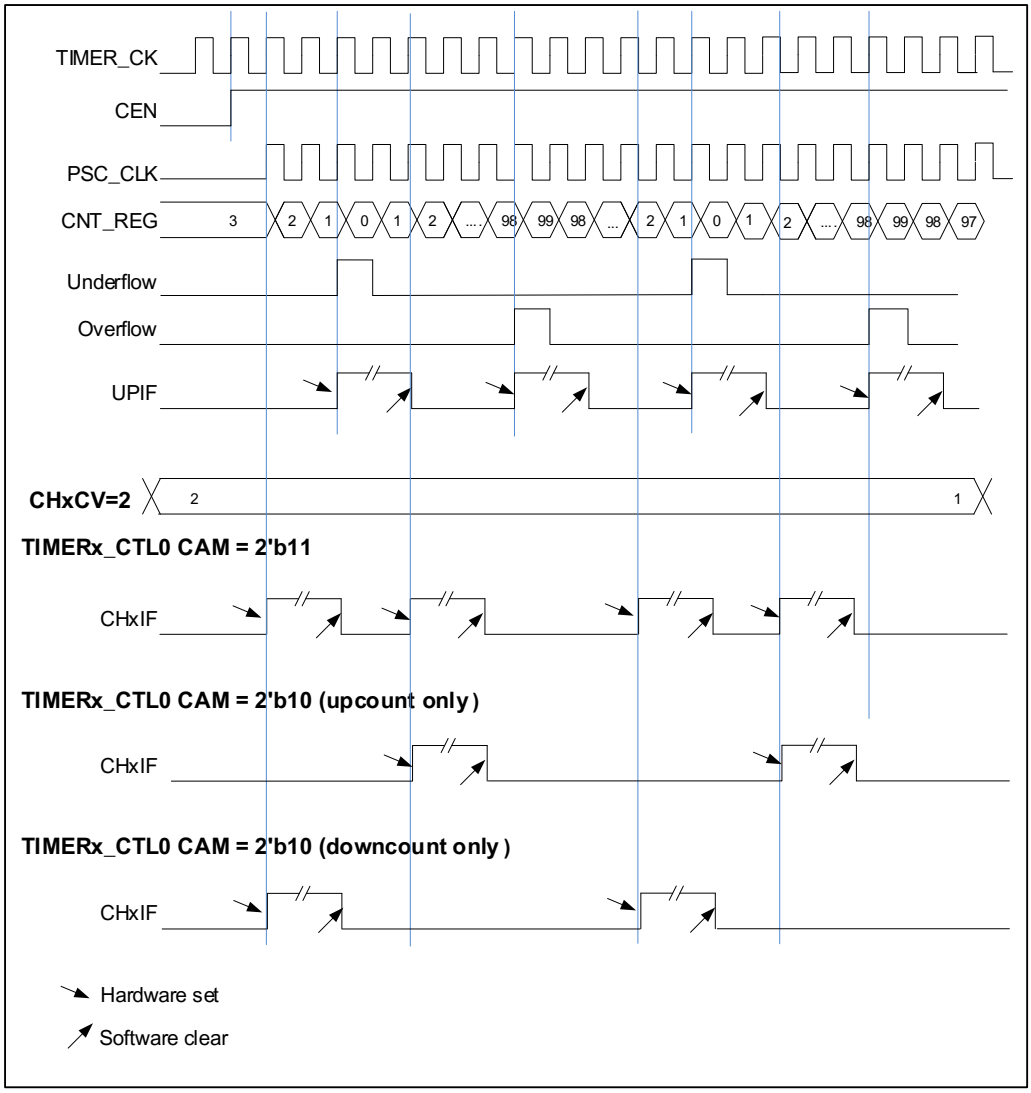
The UPIF bit in the TIMERx_SWEVG register can be set to 1 when an underflow event at count-down (CAM in TIMERx_CTL0 is "2'b01"), an overflow event at count-up (CAM in TIMERx_CTL0 is "2'b10") or both of them occur (CAM in TIMERx_CTL0 is "2'b11").

If the UPDIS bit in the TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

Figure 22-41. Timing chart of center-aligned counting mode show some examples of the counter behavior when TIMERx_CAR=0x99. TIMERx_PSC=0x0

Figure 22-41. Timing chart of center-aligned counting mode



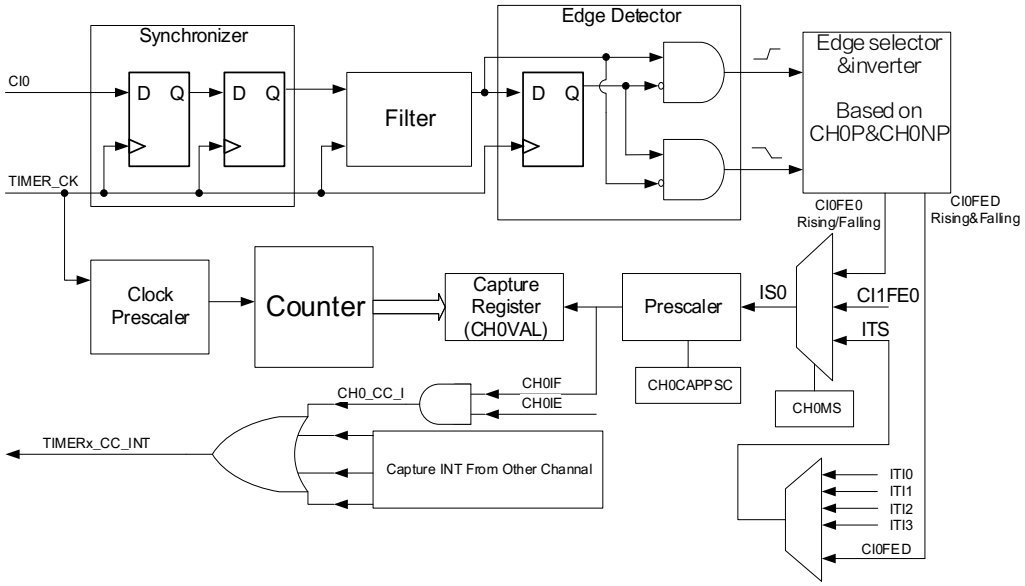
Input capture and output compare channels

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.

- **Channel input capture function**

Channel input capture function 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 enabled by CHxIE = 1.

Figure 22-42. Channel input capture principle



One of channels' input signals (CIx) can be chosen from the TIMERx_CHx signal or the Excusive-OR function of the TIMERx_CH0, TIMERx_CH1 and TIMERx_CH2 signals. First, the channel input signal (CIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

Step1: Filter Configuration. (CHxCAPFLT in TIMERx_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

Step2: Edge Selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source Selection. (CHxMS in TIMERx_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel 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 enable; you can got the interrupt and DMA request.

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: When you wanted input signal is got, TIMERx_CHxCV will be set by counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the your configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN.

Direct generation: If you want to generate a DMA request or interrupt, you can set CHxG by software directly.

The channel input capture function can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Channel output compare function**

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL 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 CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=1.

So the process can be divided to 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 high polarity by CHxP/CHxNP.
- Enable the output by CHxEN.

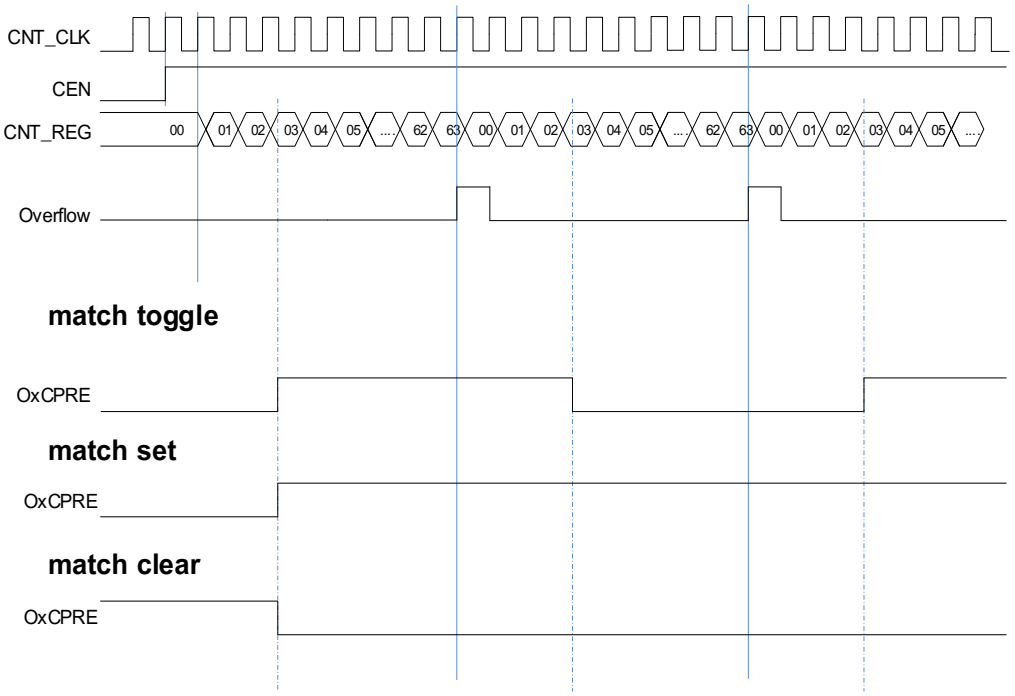
Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.
About the CHxVAL, you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 22-43. Output-compare under three modes



Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

Based on the counter mode, we have can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

The EAPWM period is determined by TIMERx_CAR and duty cycle is by TIMERx_CHxCV. ***Figure 22-44. EAPWM timechart*** 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 22-45. CAPWM timechart*** shows the CAPWM output and interrupts waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

Figure 22-44. EAPWM timechart

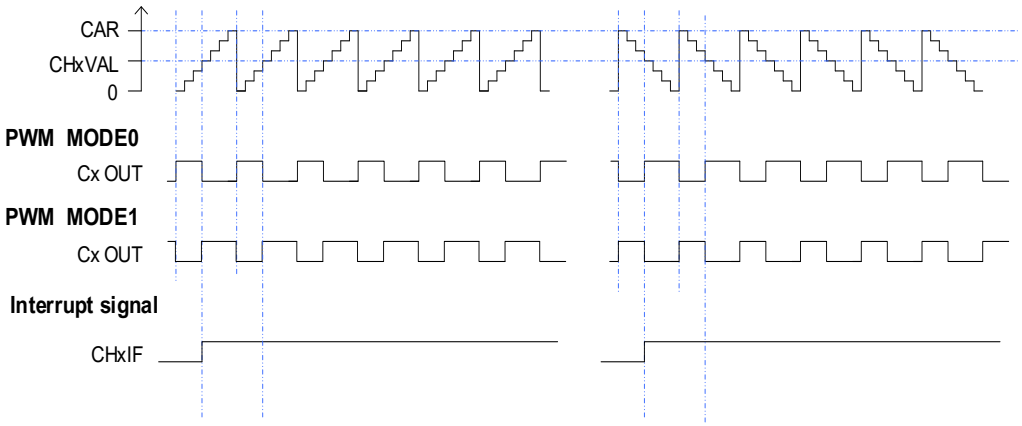
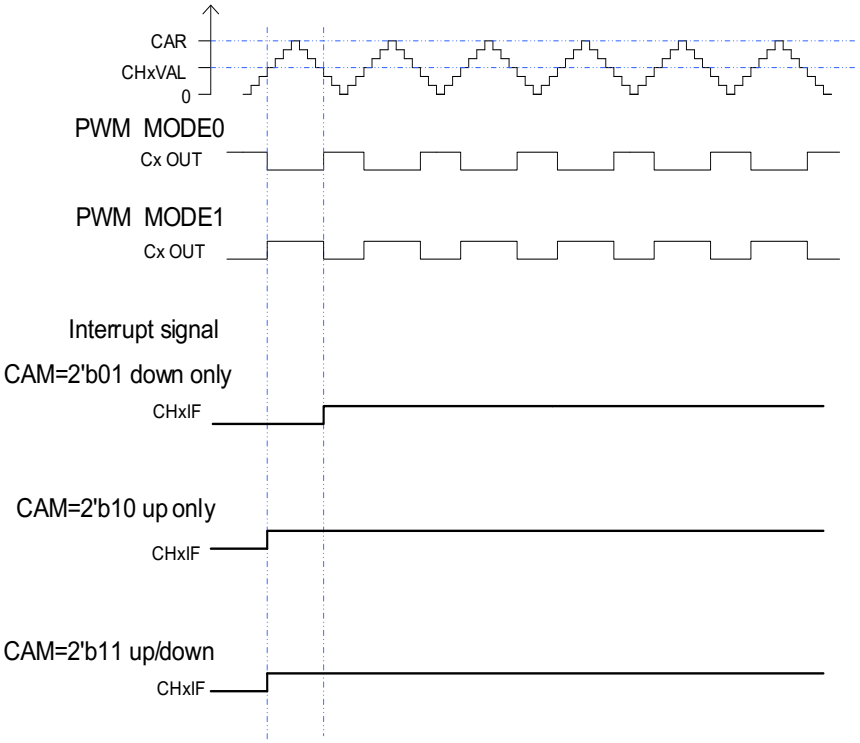


Figure 22-45. CAPWM timechart



Channel output prepare signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting 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. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

Quadrature decoder

Refer to ***Quadrature decoder***.

Hall sensor function

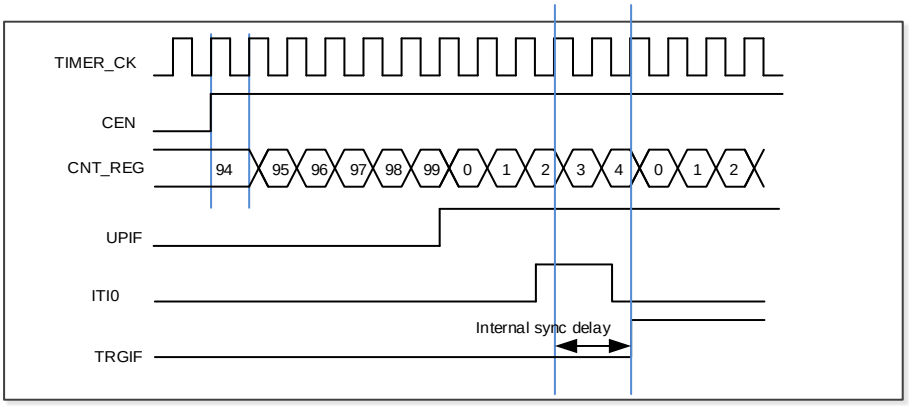
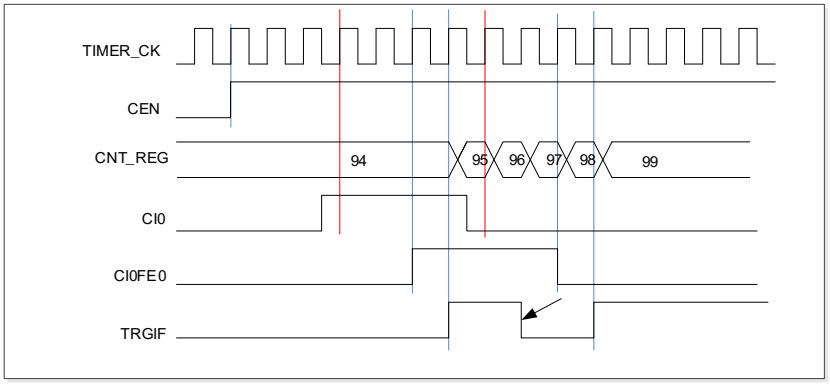
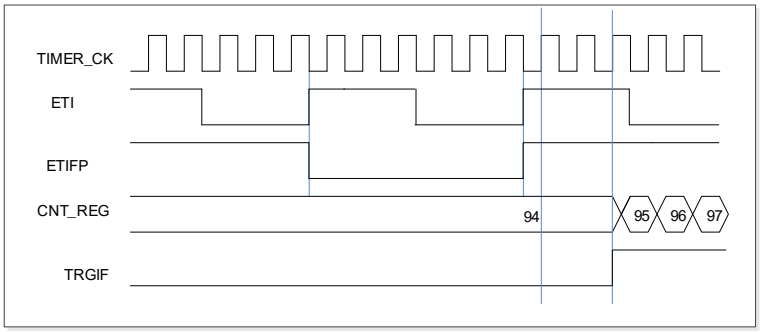
Refer to ***Hall sensor function***.

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 which is selected by the SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

Table 22-6. Slave mode examples

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion. If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000 ITI0 is the selection.	For ITI0, no polarity selector can be used.	For the ITI0, no filter and prescaler can be used.

<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 colspan="4">Figure 22-46. Restart mode

</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode
The counter can be paused when the trigger input is low.</td><td>TRGS[2:0]=3'b101
CI0FE0 is the selection.</td><td>TI0S=0(Non-xor)
[CH0NP==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 22-47. Pause mode

</td></tr>
<tr><td rowspan="2">Exam3</td><td>Event mode
The counter will start to count when a rising trigger input.</td><td>TRGS[2:0]=3'b111
ETIF is the selection.</td><td>ETP = 0 no polarity change.</td><td>ETPSC = 1, divided by 2.
ETFC = 0, no filter</td></tr>
<tr><td colspan="4">Figure 22-48. Event mode

</td></tr>
</table>

Single pulse mode

Refer to ***Single pulse mode***.

Timers interconnection

Refer to ***Advanced timer (TIMERx, x=0, 7)***.

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 interrupt 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 DMASAR+0x4, DMASAR+0x8, DMASAR+0xc at the next 3 accesses to TIMERx_DMATB. In one 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.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL2 register set to 1, the TIMERx counter stops.

22.2.4. Register definition

TIMER1 base address: 0x4000 0000

TIMER2 base address: 0x4000 0400

TIMER3 base address: 0x4000 0800

TIMER4 base address: 0x4000 0C00

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						CKDIV[1:0]		ARSE	CAM[1:0]		DIR	SPM	UPS	UPDIS	CEN
						rw		rw	rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK_TIMER}$ 01: $f_{DTS}= f_{CK_TIMER}$ /2 10: $f_{DTS}= f_{CK_TIMER}$ /4 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	Counter aligns mode selection 00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit. 01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting down, CHxF bit can be set. 10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when counting up, CHxF bit can be set. 11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in

		TIMERx_CHCTL0 register). Both when counting up and counting down, CHxF bit can be set. After the counter is enabled, cannot be switched from 0x00 to non 0x00.
4	DIR	Direction 0: Count up 1: Count down If the timer work in center-aligned mode or decoder mode, this bit is read only.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
0	CEN	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and decoder mode.

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	MMC[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 0: The TIMERx_CH0 pin input is selected as channel 0 trigger input. 1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit. 011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output. 100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE. 101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE. 110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE. 111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE.
3	DMAS	DMA request source selection 0: When capture or compare event occurs, the DMA request of channel x is sent 1: When update event occurs, the DMA request of channel x is sent.
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															

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	TRGS[2:0]			Reserved	SMC[2:0]		
rw	rw	rw		rw				rw	rw				rw		

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	ETP	External trigger polarity This bit specifies the polarity of ETI signal 0: ETI is active at rising edge or high level . 1: ETI is active at falling edge or low level .
14	SMC1	Part of SMC for enable External clock mode1. In external clock mode 1, the counter is clocked by any active edge on the ETIFP signal. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case. The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time. **Note:** External clock mode 0 enable is in this register's SMC[2:0] bit-filed.
13:12	ETPSC[1:0]	The prescaler of external trigger The frequency of external trigger signal ETIFP must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETIFP frequency. 00: Prescaler disable. 01: The prescaler is 2. 10: The prescaler is 4. 11: The prescaler is 8.
11:8	ETFC[3:0]	External trigger filter control The external trigger can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the external trigger signal

according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level.

The filtering capability configuration is as follows:

EXTFC[3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS_CK}/2$
4'b0101	8	
4'b0110	6	$f_{DTS_CK}/4$
4'b0111	8	
4'b1000	6	$f_{DTS_CK}/8$
4'b1001	8	
4'b1010	5	$f_{DTS_CK}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS_CK}/32$
4'b1110	6	
4'b1111	8	

7 MSM

Master-slave mode

This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.

0: Master-slave mode disable

1: Master-slave mode enable

6:4 TRGS[2:0]

Trigger selection

This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.

000: ITI0

001: ITI1

010: ITI2

011: ITI3

100: CI0F_ED

101: CI0FE0

110: CI1FE1

111: ETIFP

These bits must not be changed when slave mode is enabled.

3 Reserved

Must be kept at reset value.

2:0 SMC[2:0]

Slave mode control

000: Disable mode. The slave mode is disabled; The prescaler is clocked directly

by the internal clock (TIMER_CK) when CEN bit is set high.
001: Quadrature decoder mode 0.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level.
010: Quadrature decoder mode 1.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level.
011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 edge, while the direction depends on each other.
100: Restart mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input.
101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter clock when it is low.
110: Event mode. A rising edge of the trigger input enables the counter.
111: External clock mode 0. The counter counts on the rising edges of the selected trigger.

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															

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
31:15	Reserved	Must be kept at reset value.
14	TRGDEN	Trigger DMA request enable 0: disabled 1: enabled
13	Reserved	Must be kept at reset value.
12	CH3DEN	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
11	CH2DEN	Channel 2 capture/compare DMA request enable 0: disabled 1: enabled
10	CH1DEN	Channel 1 capture/compare DMA request enable 0: disabled

		1: enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5	Reserved	Must be kept at reset value.
4	CH3IE	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 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															

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	CH3IF	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
31:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag 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. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	Reserved	Must be kept at reset value.
6	TRGIF	Trigger interrupt flag This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5	Reserved	Must be kept at reset value.
4	CH3IF	Channel 3 's capture/compare interrupt enable Refer to CH0IF description
3	CH2IF	Channel 2 's capture/compare interrupt enable Refer to CH0IF description
2	CH1IF	Channel 1 's capture/compare interrupt flag Refer to CH0IF description
1	CH0IF	Channel 0 's capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 0: No Channel 1 interrupt occurred 1: Channel 1 interrupt occurred
0	UPIF	Update interrupt flag This bit is set by hardware on an update event 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															

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
31:7	Reserved	Must be kept at reset value.
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	Reserved	Must be kept at reset value.
4	CH3G	Channel 3's capture or compare event generation Refer to CH0G description
3	CH2G	Channel 2's capture or compare event generation Refer to CH0G description
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	This bit can be set by software, and cleared by hardware automatically. When this

bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.

0: No generate an update event

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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COMCEN	CH1COMCTL[2:0]			CH1COMSEN	CH1COMFEN	CH1MS[1:0]		CH0COMCEN	CH0COMCTL[2:0]			CH0COMSEN	CH0COMFEN	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
31:16	Reserved	Must be kept at reset value.
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMFEN description
9:8	CH1MS[1:0]	Channel 1 mode selection 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. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. **Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable.

		When this bit is set, if the ETIFP signal is detected as high level, the O0CPRE signal will be cleared. 0: Channel 0 output compare clear disable 1: Channel 0 output compare clear enable
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from "Timing" mode to "PWM" mode or the comparison result changes.
3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)
2	CH0COMFEN	Channel 0 output compare fast enable 0: Channel 0 output quickly compare disable.Only the valid edge of the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the result of the comparison to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable.Both the valid edge of the trigger input signal and the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the trigger input to

activate CH0_O output is 3 clock cycles.

1:0 CH0MS[1:0] Channel 0 I/O mode selection

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. (CH0EN bit in TIMERx_CHCTL2 register is reset).).

00: Channel 0 is programmed as output mode

01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0

10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0

11: Channel 0 is programmed as input mode, IS0 is connected to ITS

Note: When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 2 input capture filter control The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI2 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH2CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	

		4'b1100	8	
		4'b1101	5	f_{DTS}/32
		4'b1110	6	
		4'b1111	8	
3:2	CH0CAPPSC[1:0]	Channel 2 input capture prescaler 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 clear. 00: Prescaler disable, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on every 8 channel input edges		
1:0	CH0MS[1:0]	Channel 0 mode selection 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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3COM CEN	CH3COMCTL[2:0]			CH3COM SEN	CH3COM FEN	CH3MS[1:0]		CH2COM CEN	CH2COMCTL[2:0]			CH2COM SEN	CH2COM FEN	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
31:16	Reserved	Must be kept at reset value.
15	CH3COMCEN	Channel 3 output compare clear enable Refer to CH0COMCEN description
14:12	CH3COMCTL[2:0]	Channel 3 compare output control Refer to CH0COMCTL description
11	CH3COMSEN	Channel 3 output compare shadow enable Refer to CH0COMSEN description
10	CH3COMFEN	Channel 3 output compare fast enable Refer to CH0COMFEN description
9:8	CH3MS[1:0]	Channel 3 mode selection 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. (CH3EN bit in TIMERx_CHCTL2 register is reset).
00: Channel 3 is programmed as output mode
01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3
10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3
11: Channel 3 is programmed as input mode, IS3 is connected to ITS.
Note: When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, if the ETIFP signal is detected as high level, the O2CPRE signal will be cleared. 0: Channel 2 output compare clear disable 1: Channel 2 output compare clear enable
6:4	CH2COMCTL[2:0]	Channel 2 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits. 000: Timing mode. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT. 001: Set the channel output. O2CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV. 100: Force low. O2CPRE is forced to low level. 101: Force high. O2CPRE is forced to high level. 110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low when the counter is larger than TIMERx_CH2CV, and high otherwise. 111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller than TIMERx_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx_CH2CV, and low otherwise. If configured in PWM mode, the O2CPRE level changes only when the output compare mode is adjusted from "Timing" mode to "PWM" mode or the comparison result changes.
3	CH2COMSEN	Channel 2 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled. 0: Channel 2 output compare shadow disable 1: Channel 2 output compare shadow enable The PWM mode can be used without verifying the shadow register only in single

pulse mode (when SPM=1)

2	CH2COMFEN	Channel 2 output compare fast enable 0: Channel 2 output quickly compare disable.Only the valid edge of the comparison result will generate a comparison match, and CH2_O is set to the compare level.The minimum delay from an edge on the result of the comparison to activate CH2_O output is 5 clock cycles. 1: Channel 2 output quickly compare enable.Both the valid edge of the trigger input signal and the comparison result will generate a comparison match, and CH2_O is set to the compare level.The minimum delay from an edge on the trigger input to activate CH2_O output is 3 clock cycles.
1:0	CH2MS[1:0]	Channel 2 I/O mode selection 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. (CH2EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 2 is programmed as output mode 01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2 10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2 11: Channel 2 is programmed as input mode, IS2 is connected to ITS. **Note:** When CH2MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:12	CH3CAPFLT[3:0]	Channel 3 input capture filter control Refer to CH0CAPFLT description
11:10	CH3CAPPSC[1:0]	Channel 3 input capture prescaler Refer to CH0CAPPSC description
9:8	CH3MS[1:0]	Channel 3 mode selection Same as Output compare mode
7:4	CH2CAPFLT[3:0]	Channel 2 input capture filter control The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI2 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH2CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}

4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	8	

3:2 CH2CAPPSC[1:0] Channel 2 input capture prescaler

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 clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH2MS[1:0] Channel 2 mode selection

Same as output compare mode

Channel control register 2 (TIMERx_CHCTL2)

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		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

Bits	Fields	Descriptions
31:14	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

		Refer to CH0EN description
11	CH2NP	Channel 2 complementary output polarity Refer to CH0NP description
10	Reserved	Must be kept at reset value.
9	CH2P	Channel 2 capture/compare function polarity Refer to CH0P description
8	CH2EN	Channel 2 capture/compare function enable Refer to CH0EN description
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	Reserved	Must be kept at reset value.
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit should be keep reset value. When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	Reserved	Must be kept at reset value.
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted. 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 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. 0: Channel 0 disabled 1: Channel 0 enabled

Counter register (TIMERx_CNT) (x=1, 4)

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[31:16]															
rw															
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[31:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

Counter register (TIMERx_CNT) (x=2, 3)

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
CNT[15:0]															
rw															

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. Writing to this bit-filed 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 The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

Counter auto reload register (TIMERx_CAR) (x=1, 4)

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 This bit-filed specifies the auto reload value of the counter.

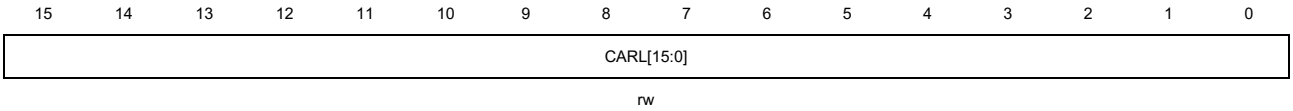
Counter auto reload register (TIMERx_CAR) (x=2, 3)

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															



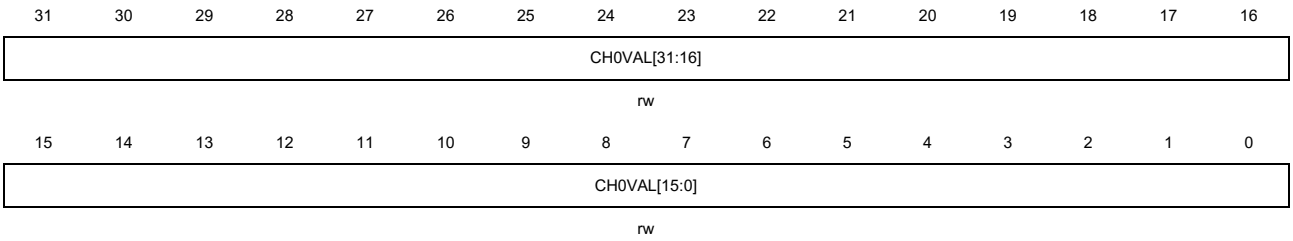
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

Channel 0 capture/compare value register (TIMERx_CH0CV) (x=1, 4)

Address offset: 0x34

Reset value: 0x0000 0000

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



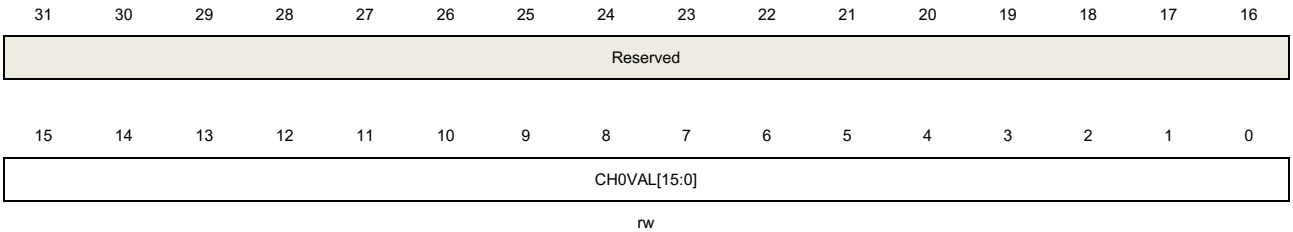
Bits	Fields	Descriptions
31:0	CH0VAL[31:0]	Capture or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 0 capture/compare value register (TIMERx_CH0CV) (x=2, 3)

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 or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 1 capture/compare value register (TIMERx_CH1CV) (x=1, 4)

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
CH1VAL[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:0	CH1VAL[31:0]	Capture or compare value of channel1 When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 1 capture/compare value register (TIMERx_CH1CV) (x=2, 3)

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
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.

15:0	CH1VAL[15:0]	Capture or compare value of channel1 When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 2 capture/compare value register (TIMERx_CH2CV) (x=1, 4)

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
31:0	CH2VAL[31:0]	Capture or compare value of channel 2 When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 2 capture/compare value register (TIMERx_CH2CV) (x=2, 3)

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
CH2VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH2VAL[15:0]	Capture or compare value of channel 2 When channel 2 is configured in input mode, this bit-filed indicates the counter value

corresponding to the last capture event. And this bit-filed is read-only.
When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 3 capture/compare value register (TIMERx_CH3CV) (x=1, 4)

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
31:0	CH3VAL[31:0]	Capture or compare value of channel 3 When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Channel 3 capture/compare value register (TIMERx_CH3CV) (x=2, 3)

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
CH3VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH3VAL[15:0]	Capture or compare value of channel 3 When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 3 is configured in output mode, this bit-filed contains value to be

compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

DMA configuration register (TIMERx_DMACFG)

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															

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

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001.
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.

DMA transfer buffer register (TIMERx_DMATB)

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
Reserved															

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.

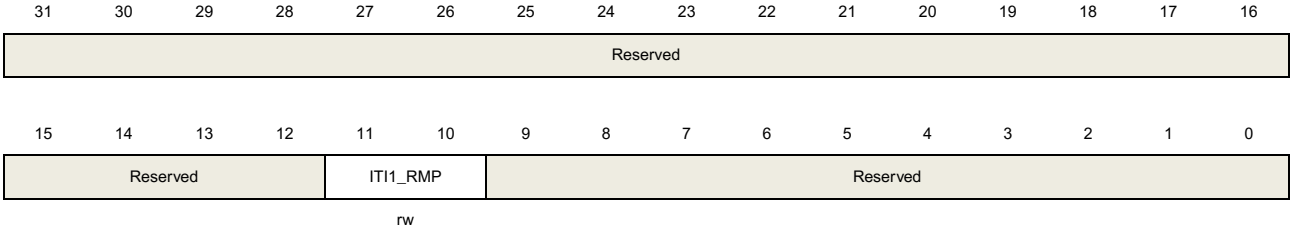
15:0	DMATB[15:0]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

Input remap register (TIMERx_IRMP) (x=1)

Address offset: 0x50

Reset value: 0x0000 0000

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



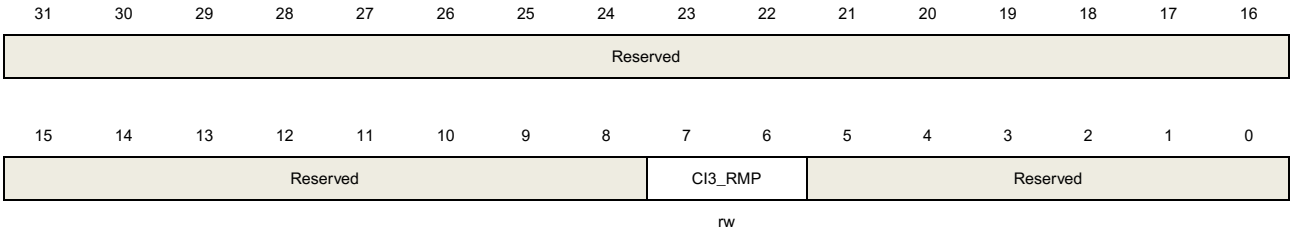
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:10	ITI1_RMP	Internal trigger input1 remap 00:TIMER7_TRGO 01:Ethernet PTP 10:USB FS SOF 11:USB HS SOF
9:0	Reserved	Must be kept at reset value.

Input remap register (TIMERx_IRMP) (x=4)

Address offset: 0x50

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.

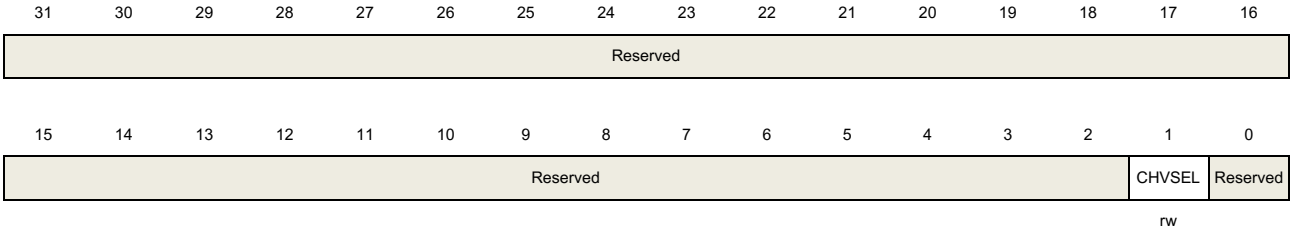
7:6	CI3_RMP	Channel 3 input remap. 00: GPIO pin. Refer to GPIO remap table 01: IRC32K 10: LXTAL 11:RTC wakeup interrupt
5:0	Reserved	Must be kept at reset value.

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 This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

22.3. General level1 timer (TIMERx, x=8, 11)

22.3.1. Overview

The general level1 timer module (Timer8, 11) is a two-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level1 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level1 timers can be programmed and be used to count or time external events that drive other Timers.

Timer and timer are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

22.3.2. Characteristics

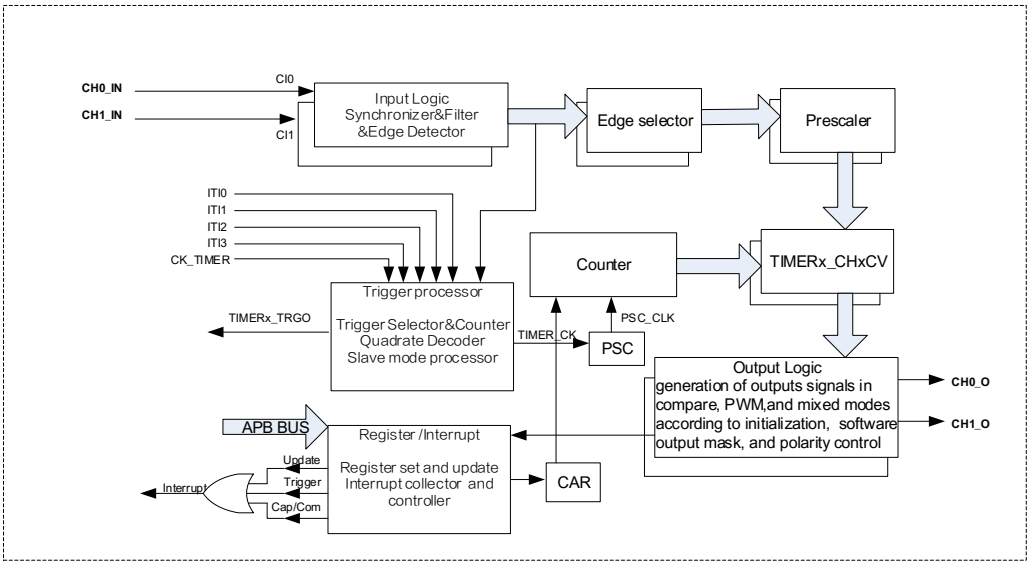
- Total channel num: 2.
- Counter width: 16bit.
- Source of count clock is selectable:
  internal clock, internal trigger, external input, external trigger.
- counter mode: Count up only.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:
  Input capture mode, Output compare mode, Programmable PWM mode, Single pulse mode
- Auto-reload function.
- Interrupt output on: update, trigger event, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer master-slave management.

22.3.3. Function overview

Block diagram

Figure 22-49. General level1 timer block diagram provides details on the internal configuration of the general level1 timer.

Figure 22-49. General level1 timer block diagram



Clock source configuration

The general level1 TIMER has the capability of being clocked by either the CK_TIMER or an alternate clock source controlled by SMC (TIMERx_SMCFG bit [2:0]).

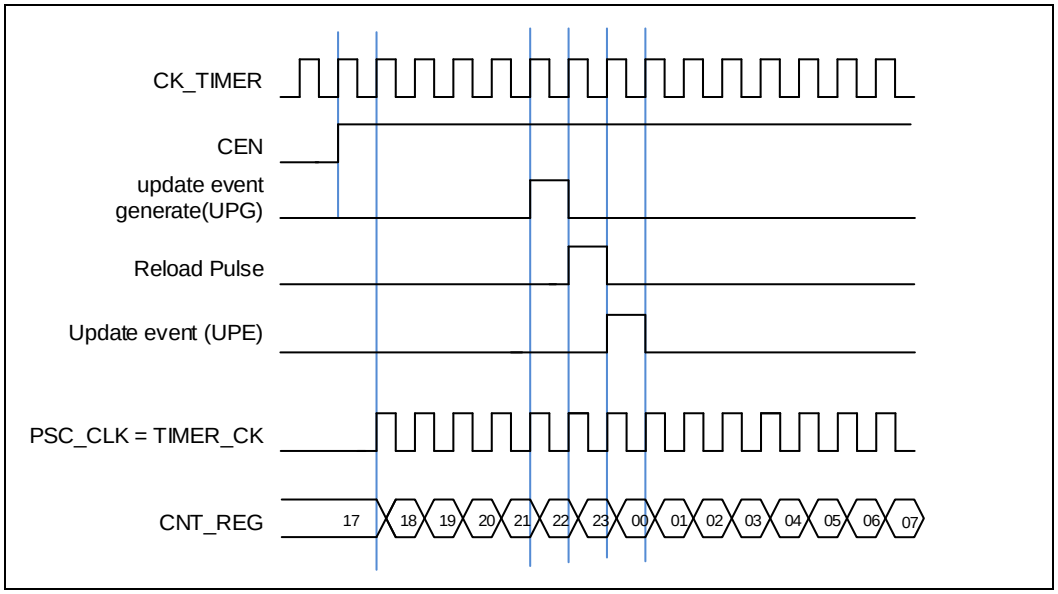
- SMC [2:0] == 3'b000. Internal timer clock CK_TIMER which is from module RCU.

The default internal clock source is the CK_TIMER used to drive the counter prescaler when the SMC [2:0] == 3'b000. When the CEN is set, the CK_TIMER will be divided by PSC value to generate PSC_CLK.

In this mode, the TIMER_CK, driven counter's prescaler to count, is equal to CK_TIMER which is from RCU.

If the SMC [2:0] in the TIMERx_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx_SMCFG register and described as follows. When the SMC bits are set to 0x4, 0x5 or 0x6, the internal clock CK_TIMER is the counter prescaler driving clock source.

Figure 22-50. Timing chart of internal clock divided by 1



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin source

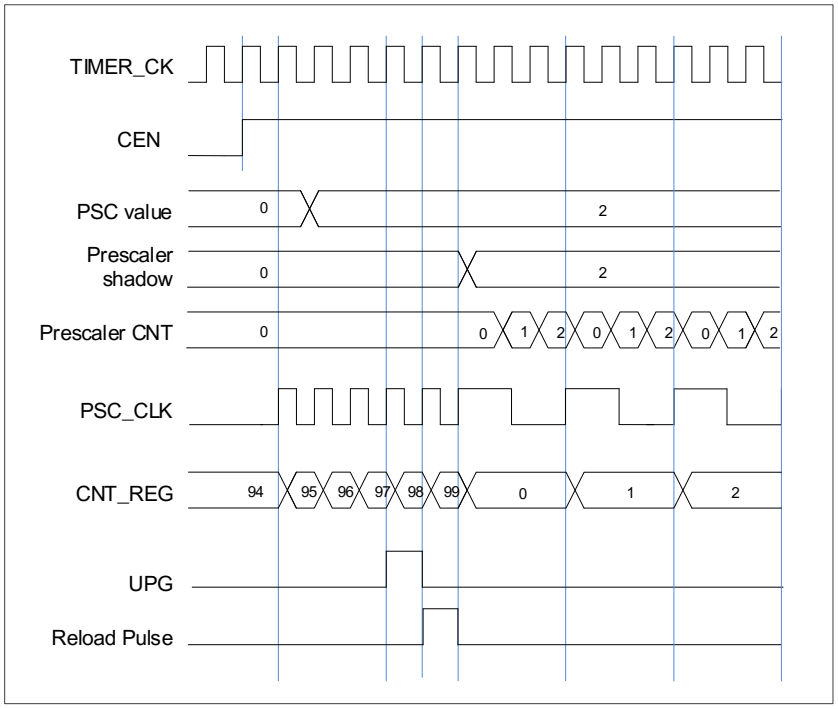
The TIMER_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx_CI0/TIMERx_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

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 SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 22-51. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. The update event is generated at each counter overflow.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter 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.

Figure 22-52. Timing chart of up counting mode, PSC=0/2

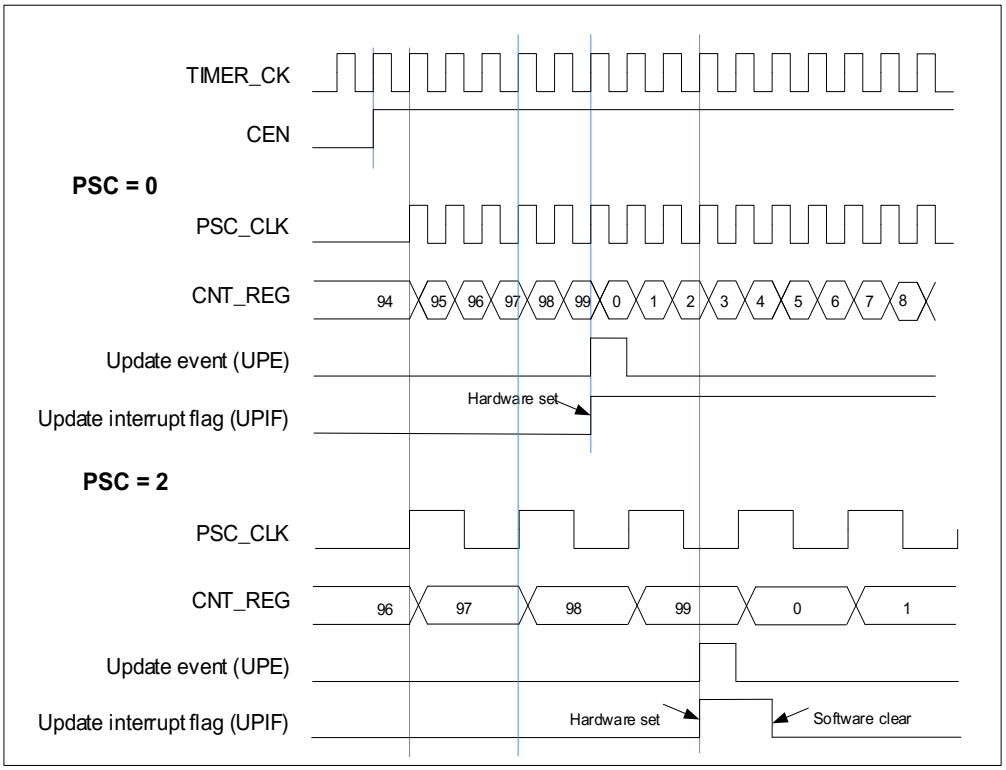
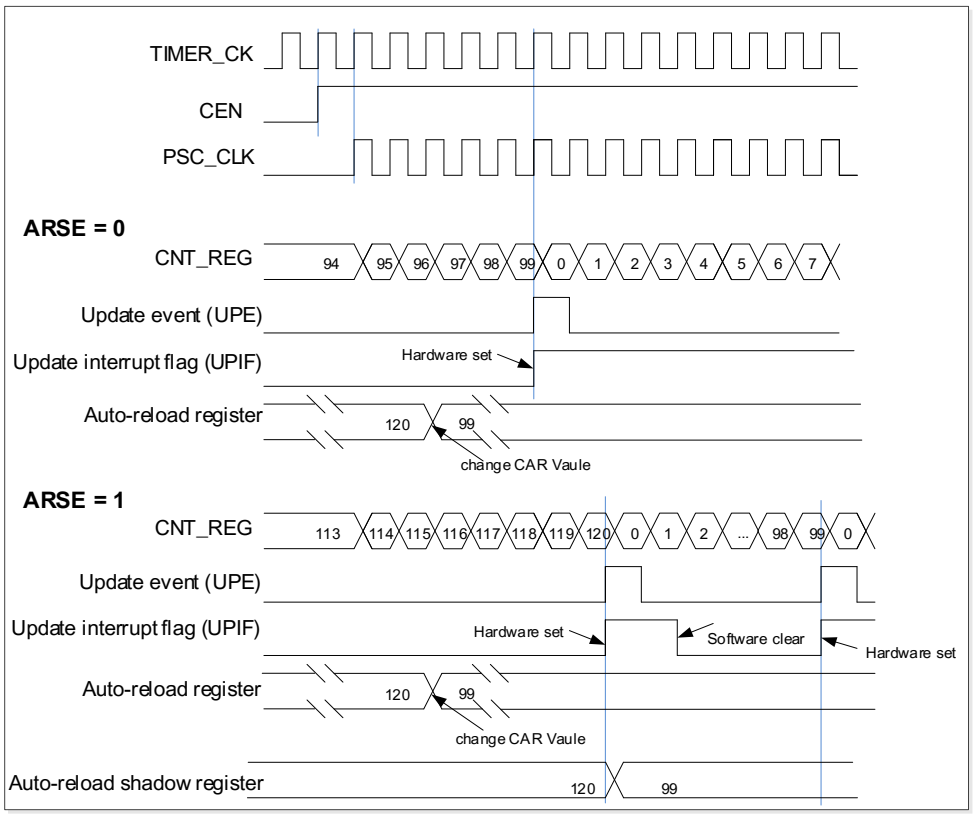


Figure 22-53. Timing chart of up counting mode, change TIMERx_CAR ongoing



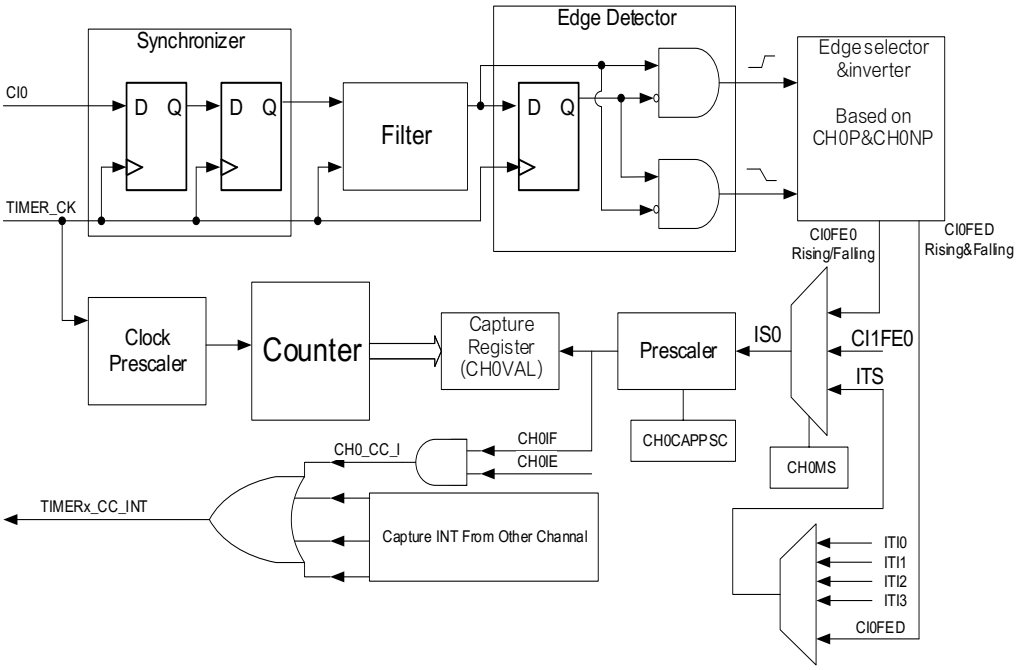
Input capture and output compare channels

The general level1 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.

- **Channel input capture function**

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 enabled by CHxIE = 1.

Figure 22-54. Channel input capture principle



First, the channel input signal (CIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

Step1: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

Step2: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source selection. (CHxMS in TIMERx_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel 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 enable; you can got the interrupt and DMA request.

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: When you wanted input signal is got, TIMERx_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the your configuration of CHxIE and CHxDEN in TIMERx_DMAINTEN.

Direct generation: If you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERX_CH0CV can measure the PWM period and the TIMERx_CH1CV can measure the PWM duty.

- **Channel output compare function**

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL 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 CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=1.

So the process can be divided to 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 high polarity by CHxP/CHxNP.
- Enable the output by CHxEN.

Step3: Interrupt/DMA-request enables configuration by CHxIE/CxCDE

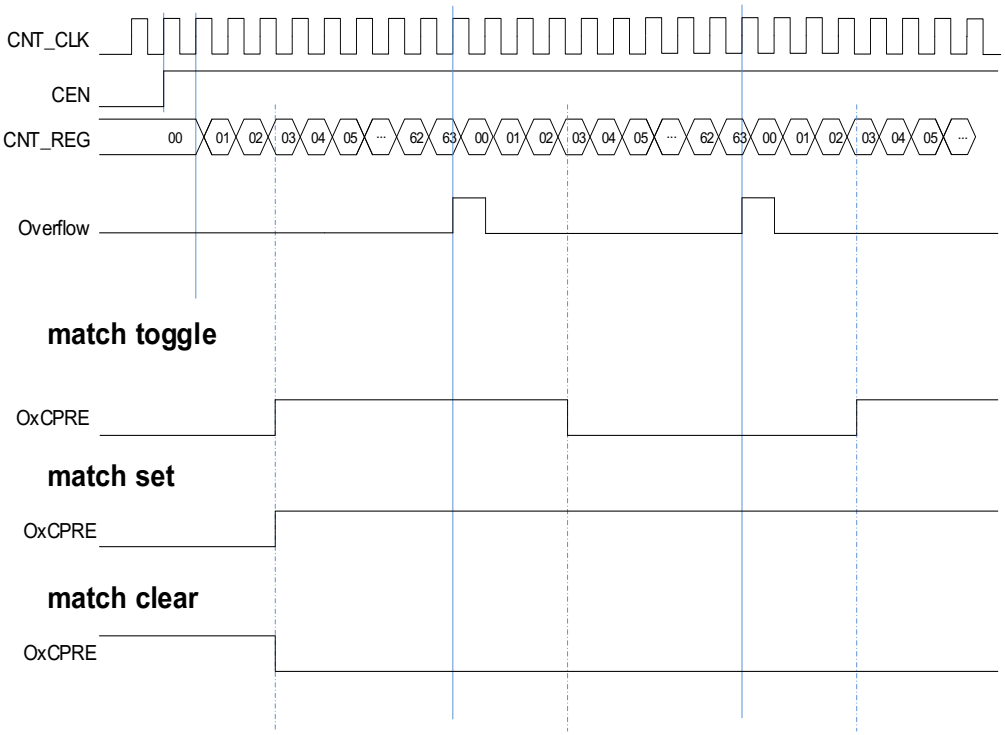
Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 22-55. Output-compare under three modes



Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b111 (PWM mode1), the channel can outputs PWM waveform according to the TIMERx_CAR registers and TIMERx_CHxCV registers.

Based on the counter mode, we have can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

The EAPWM period is determined by TIMERx_CAR and duty cycle is by TIMERx_CHxCV. ***Figure 22-56. EAPWM timechart*** 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 22-57. CAPWM timechart*** shows the CAPWM output and interrupt waveform.

If TIMERx_CHxCV is greater than TIMERx_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).
And if TIMERx_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

Figure 22-56. EAPWM timechart

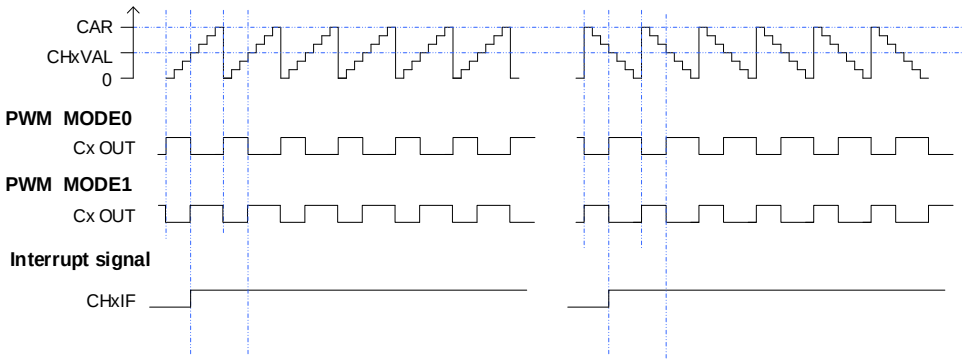
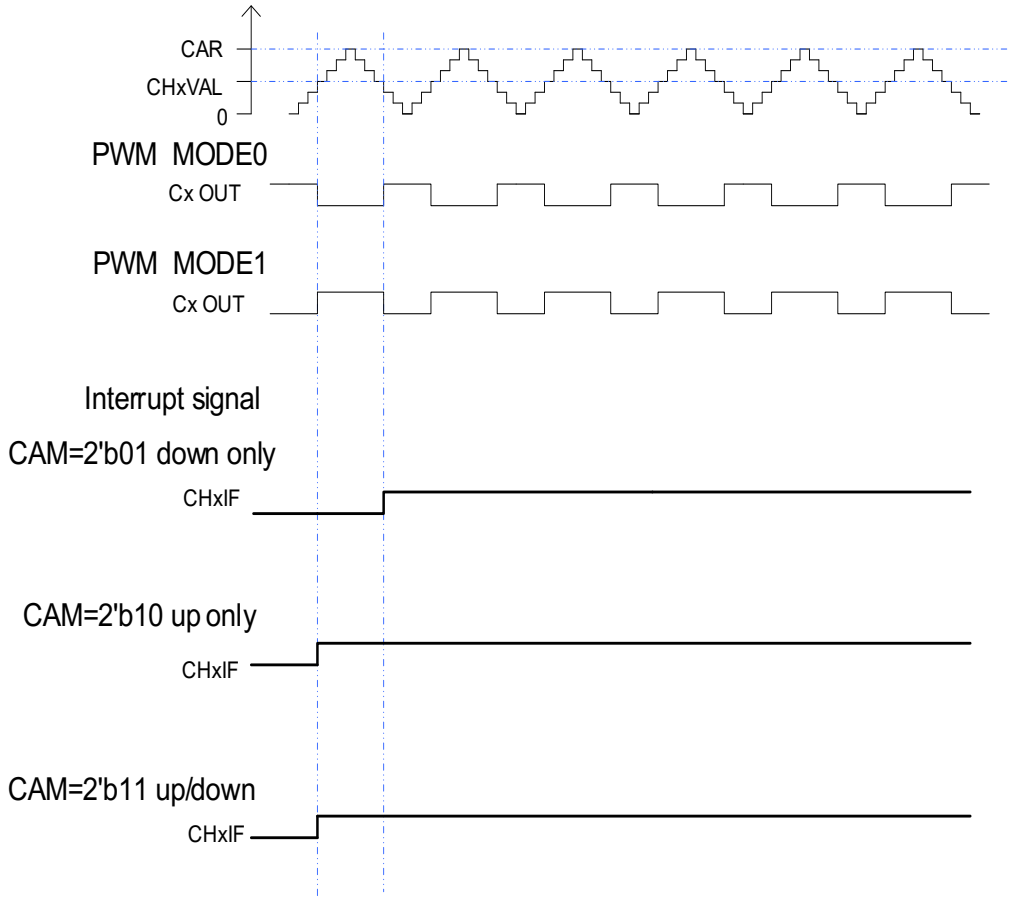


Figure 22-57. CAPWM timechart



Channel output prepare signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to

0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting 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. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx_CHxCV values.

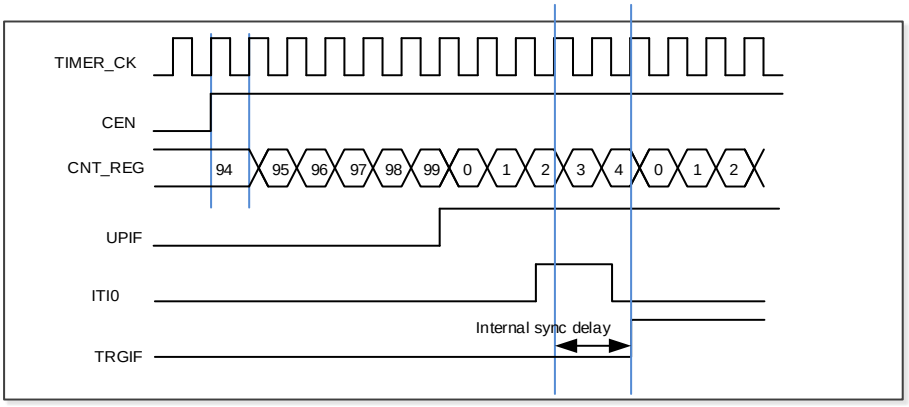
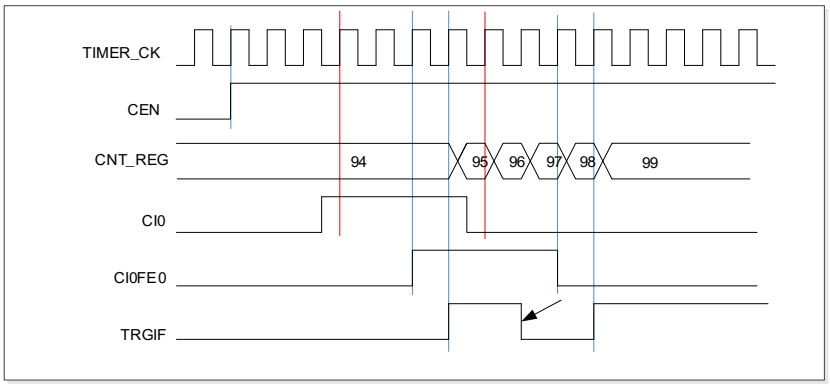
The OxCPRE signal can be forced to 0 when the ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

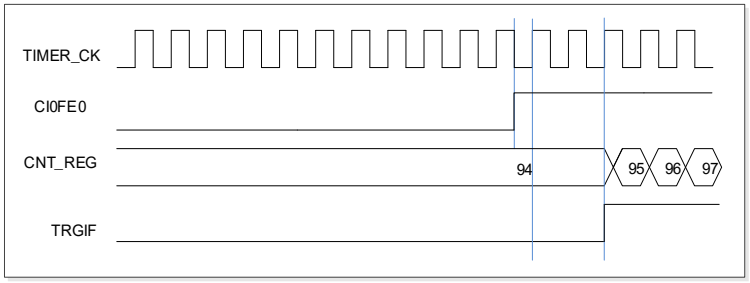
Master-slave management

The TIMERx can be synchronized with a trigger in several modes including the Restart mode, the Pause mode and the Event mode which is selected by the SMC [2:0] in the TIMERx_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx_SMCFG register.

Table 22-7. Slave mode examples

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion. If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000 ITI0 is the selection.	For ITI0, no polarity selector can be used.	For the ITI0, no filter and prescaler can be used.

<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 colspan="4">Figure 22-58. Restart mode

</td></tr>
<tr><td rowspan="2">Exam2</td><td>Pause mode
The counter can be paused when the trigger input is low.</td><td>TRGS[2:0]=3'b101
CI0FE0 is the selection.</td><td>TI0S=0(Non-xor)
[CH0NP==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 22-59. Pause mode

</td></tr>
<tr><td>Exam3</td><td>Event mode
The counter will start to count when a rising trigger input.</td><td>TRGS[2:0]=3'b101
CI0FE0 is selected.</td><td>CH0P=0,
CI0FE0 does not invert. The capture event will occur on the rising edge only.</td><td>Filter is bypassed in this example.</td></tr>
</table>

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	Figure 22-60. Event mode 			

Single pulse mode

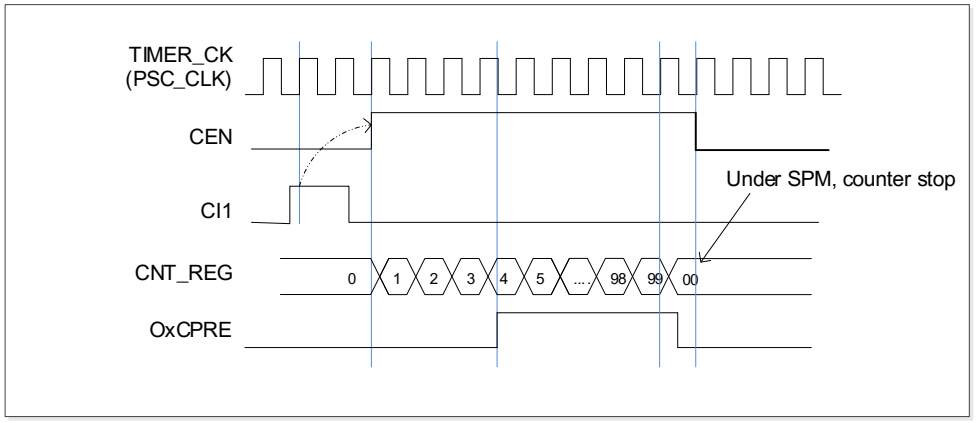
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals 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 using software, the counter will be stopped and its value held.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMERx_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMERx_CHCTL0/1 register. 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. The CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

Figure 22-61. Single pulse mode TIMERx CHxCV = 4 TIMERx CAR=99 shows an example.

Figure 22-61. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99



Timers interconnection

Refer to ***Advanced timer (TIMERx, x=0, 7)***.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL2 register set to 1, the TIMERx counter stops.

22.3.4. Register definition

TIMER8 base address: 0x4001 4000

TIMER11 base address: 0x4000 1800

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						CKDIV[1:0]		ARSE	Reserved			SPM	UPS	UPDIS	CEN
						rw		rw				rw	rw	rw	rw

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK_TIMER}$ 01: $f_{DTS}= f_{CK_TIMER}$ /2 10: $f_{DTS}= f_{CK_TIMER}$ /4 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:4	Reserved	Must be kept at reset value.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event

1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
0	CEN	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and decoder mode.

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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								MSM	TRGS[2:0]			Reserved	SMC[2:0]		
								rw	rw				rw		

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	MSM	Master-slave mode This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together. 0: Master-slave mode disable 1: Master-slave mode enable
6:4	TRGS[2:0]	Trigger selection This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter. 000: ITI0 001: ITI1

		010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: Reserved. These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	Slave mode control 000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high. 001: Reserved 010: Reserved 011: Reserved 100: Restart mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input. 101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter when it is low. 110: Event mode. A rising edge of the trigger input enables the counter. 111: External clock mode0. The counter counts on the rising edges of the selected trigger.

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									TRGIE	Reserved			CH1IE	CH0IE	UPIE
									rw				rw	rw	rw

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5:3	Reserved	Must be kept at reset value.
2	CH1IE	Channel 1 capture/compare interrupt enable

		0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CH1OF	CH0OF	Reserved		TRGIF	Reserved			CH1IF	CH0IF	UPIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag 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. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	Reserved	Must be kept at reset value.
6	TRGIF	Trigger interrupt flag This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5:3	Reserved	Must be kept at reset value.
2	CH1IF	Channel 1 's capture/compare interrupt flag

		Refer to CH0IF description
1	CH0IF	Channel 0 's capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 0: No Channel 1 interrupt occurred 1: Channel 1 interrupt occurred
0	UPIF	Update interrupt flag This bit is set by hardware on an update event 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TRGG	Reserved.			CH1G	CH0G	UPG
									w				w	w	w

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5:3	Reserved	Must be kept at reset value.
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in

		TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time. 0: No generate an update event 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
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CH1COMCTL[2:0]			CH1COMSEN	CH1COMFEN	CH1MS[1:0]		Reserved	CH0COMCTL[2:0]			CH0COMSEN	CH0COMFEN	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
31:15	Reserved	Must be kept at reset value.
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMFEN description
9:8	CH1MS[1:0]	Channel 1 mode selection 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. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. **Note:** When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise. If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from "Timing" mode to "PWM" mode or the comparison result changes.
3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)
2	CH0COMFEN	Channel 0 output compare fast enable 0: Channel 0 output quickly compare disable.Only the valid edge of the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the result of the comparison to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable.Both the valid edge of the trigger input signal and the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.
1:0	CH0MS[1:0]	Channel 0 I/O mode selection 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).).
00: Channel 0 is programmed as output mode
01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0
10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0
11: Channel 0 is programmed as input mode, IS0 is connected to ITS
Note: When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.

Input capture mode:

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture filter control The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH0CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	

		4'b1111	8	

3:2	CH0CAPPSC[1:0]	Channel 0 input capture prescaler 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 clear. 00: Prescaler disable, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on every 8 channel input edges
1:0	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

Channel control register 2 (TIMERx_CHCTL2)

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								CH1NP	Reserved	CH1P	CH1EN	CH0NP	Reserved	CH0P	CH0EN
								rw		rw	rw	rw		rw	rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	Reserved	Must be kept at reset value.
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH1EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit should be keep reset value. When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	Reserved	Must be kept at reset value.
1	CH0P	Channel 0 capture/compare function polarity

		When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted. 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 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. 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
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]	This bit-filed indicates the current counter value. Writing to this bit-filed 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 The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed 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 This bit-filed 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).

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 or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates 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).

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
CH1VAL[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CH1VAL[15:0]	Capture or compare value of channel1 When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

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	CHVSEL	Reserved
	rw	rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

22.4. General level2 timer (TIMERx, x=9, 10, 12, 13)

22.4.1. Overview

The general level2 timer module (Timer9, 10, 12, 13) is a one-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level2 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level2 timers can be programmed and be used to count or time external events that drive other Timers.

22.4.2. Characteristics

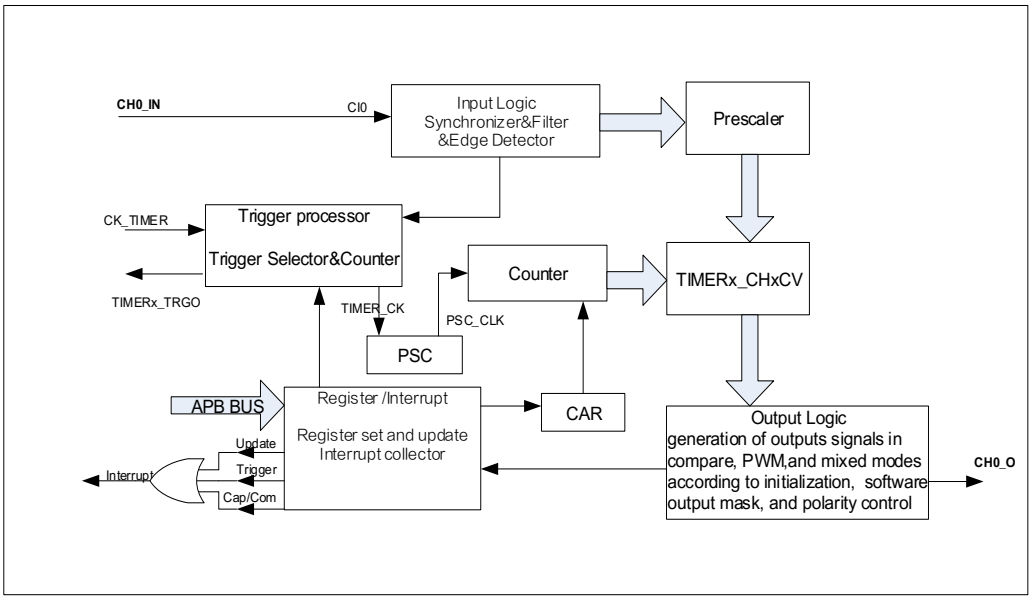
- Total channel num: 1.
- Counter width: 16bit.
- Source of count clock is internal clock only.
- Counter mode: count up only.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:
  Input capture mode, output compare mode, programmable and PWM mode.
- Auto-reload function.
- Interrupt output on: update, trigger event, and compare/capture event.

22.4.3. Function overview

Block diagram

Figure 22-62. General level2 timer block diagram provides details on the internal configuration of the general level2 timer.

Figure 22-62. General level2 timer block diagram



Clock source configuration

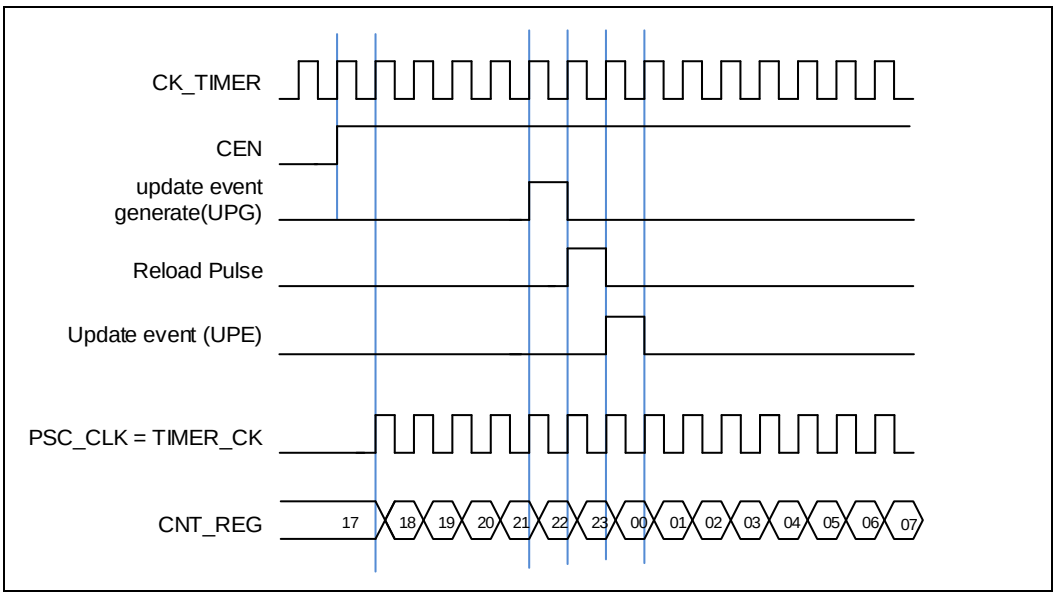
The general level2 TIMER can only being clocked by the CK_TIMER.

- Internal timer clock CK_TIMER which is from module RCU

The general level2 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

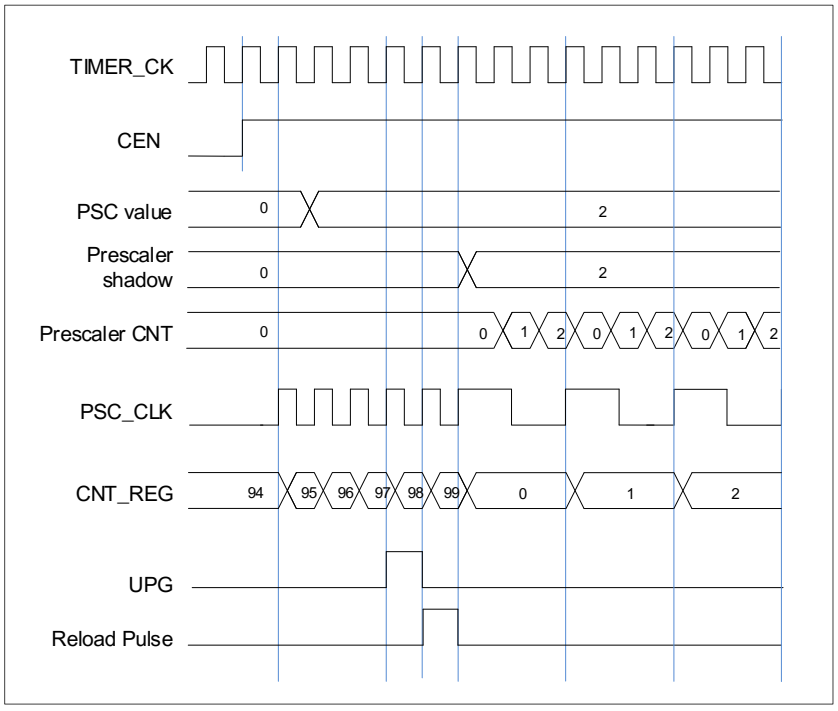
Figure 22-63. Timing chart of internal clock divided by 1



Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 22-64. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. The update event is generated at each counter overflow.

When the update event is set by the UPG bit in the TIMERx_SWEVG register, the counter value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMERx_CTL0 register is set, the update event is disabled.

When an update event occurs, all the shadow registers (counter 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.

Figure 22-65. Timing chart of up counting mode, PSC=0/2

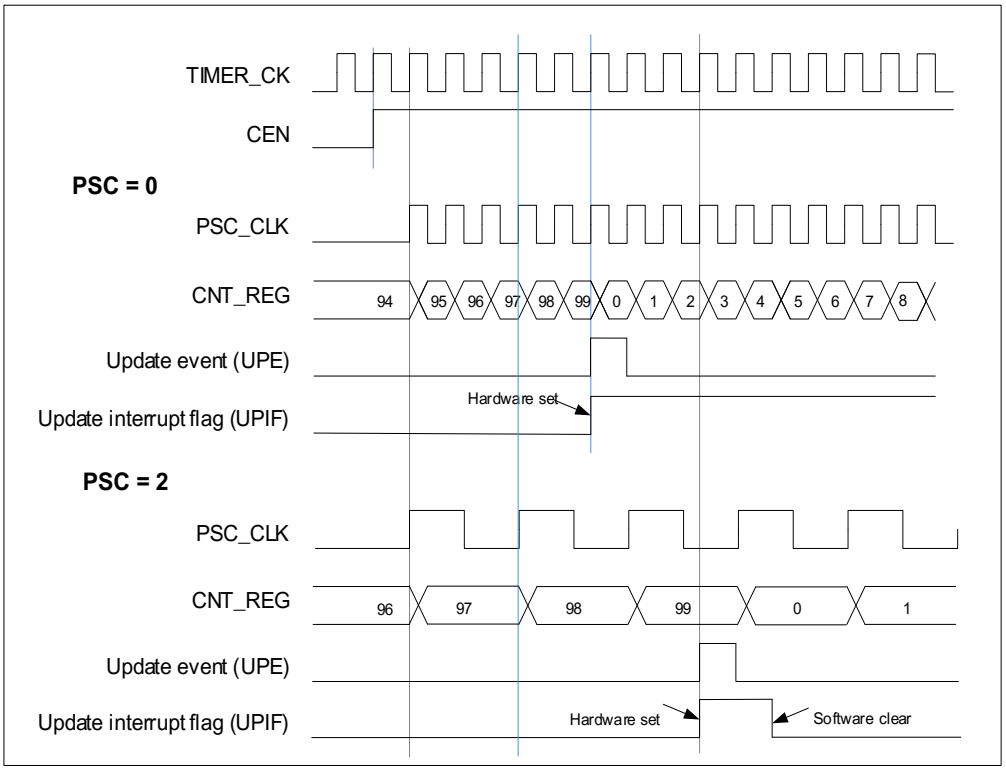
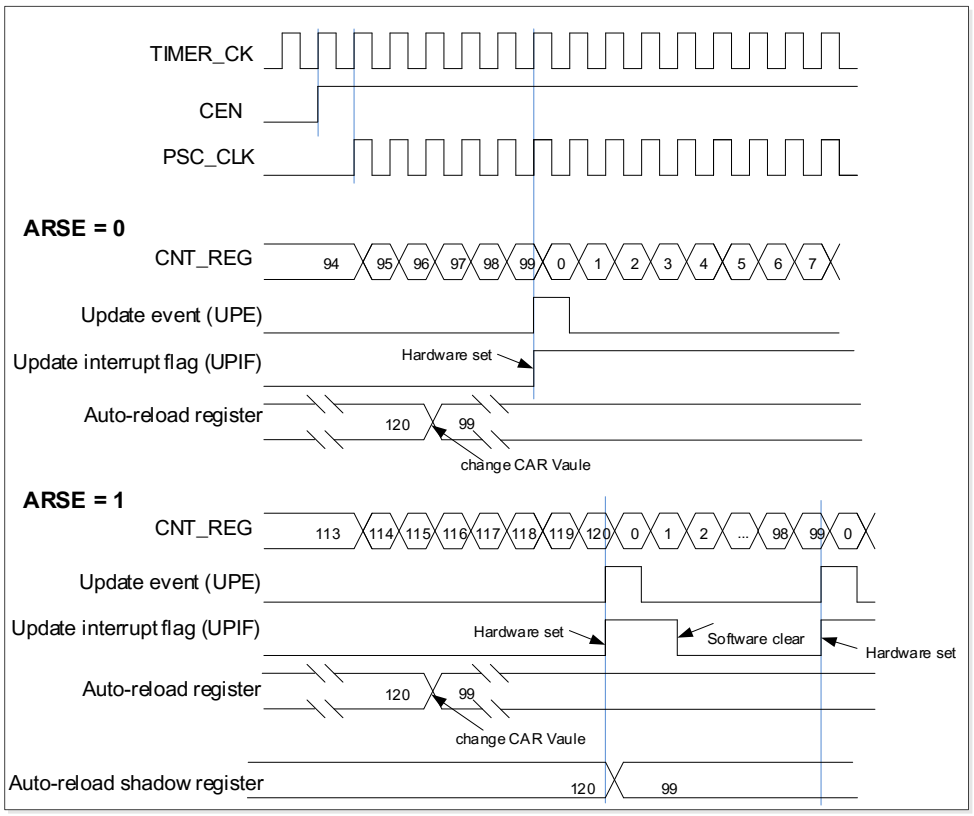


Figure 22-66. Timing chart of up counting mode, change TIMERx_CAR ongoing



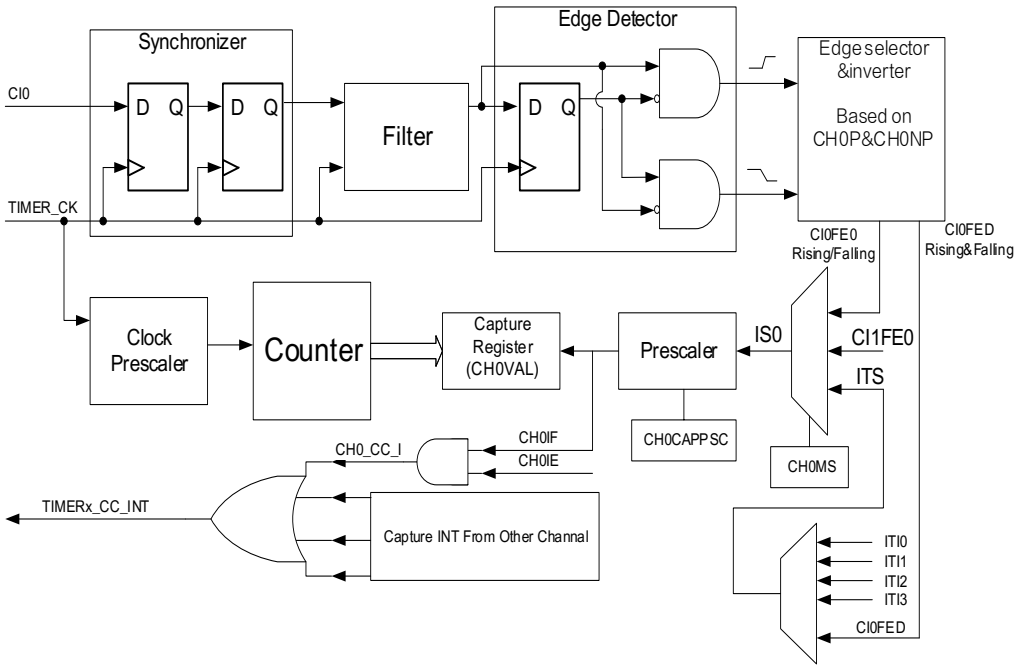
Input capture and output compare channels

The general level2 timer has one independent channel 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.

- **Channel input capture function**

Channel input capture function 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 enabled by CHxIE = 1.

Figure 22-67. Channel input capture principle



First, the channel input signal (CIx) is synchronized to TIMER_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. The IC_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

Step1: Filter configuration. (CHxCAPFLT in TIMERx_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

Step2: Edge selection. (CHxP/CHxNP in TIMERx_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

Step3: Capture source selection. (CHxMS in TIMERx_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS!=0x0) and TIMERx_CHxCV cannot be written any more.

Step4: Interrupt enable. (CHxIE in TIMERx_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt.

Step5: Capture enables. (CHxEN in TIMERx_CHCTL2)

Result: When you wanted input signal is got, TIMERx_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt will be asserted based on the your configuration of CHxIE in TIMERx_DMAINTEN.

Direct generation: If you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

- **Channel output compare function**

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL 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 CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1.

So the process can be divided to 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 high polarity by CHxP/CHxNP.
- Enable the output by CHxEN.

Step3: Interrupt/DMA-request enables configuration by CHxIE

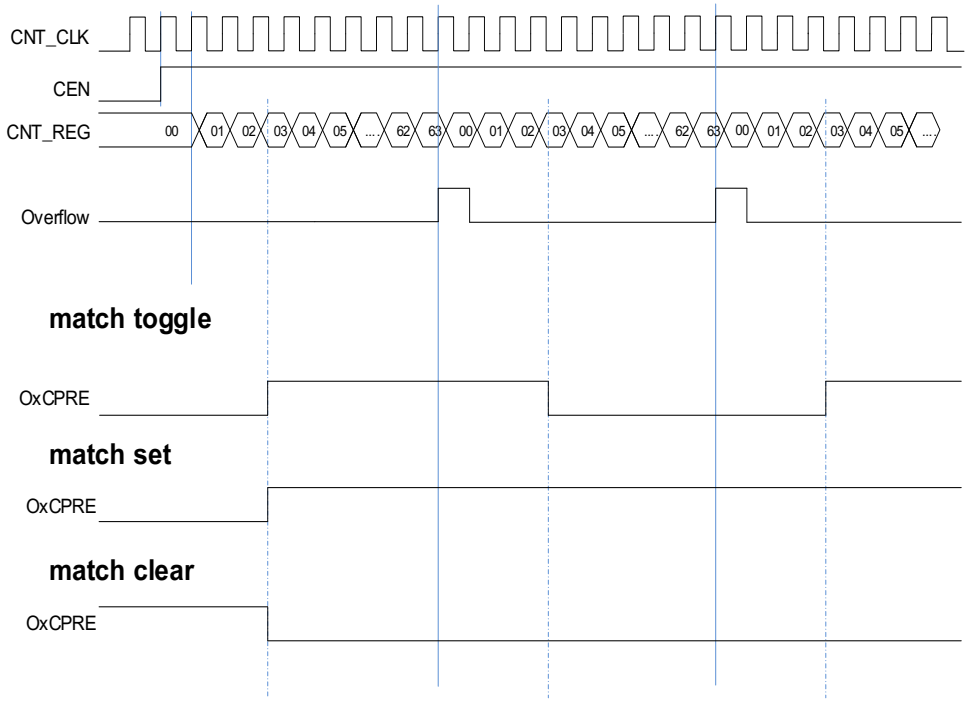
Step4: Compare output timing configuration by TIMERx_CAR and TIMERx_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

Step5: Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 22-68. Output-compare under three modes



Channel output prepare signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting 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. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

Timers interconnection

Refer to ***Advanced timer (TIMERx, x=0, 7)***.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL2 register set to 1, the TIMERx counter stops.

22.4.4. Register definition

TIMER9 base address: 0x4001 4400

TIMER10 base address: 0x4001 4800

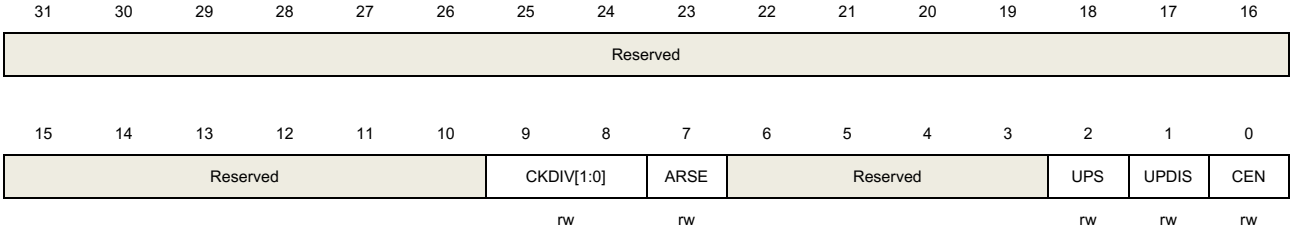
TIMER12 base address: 0x4000 1C00

TIMER13 base address: 0x4000 2000

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
31:10	Reserved	Must be kept at reset value.
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK_TIMER}$ 01: $f_{DTS}= f_{CK_TIMER}$ /2 10: $f_{DTS}= f_{CK_TIMER}$ /4 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:3	Reserved	Must be kept at reset value.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable.

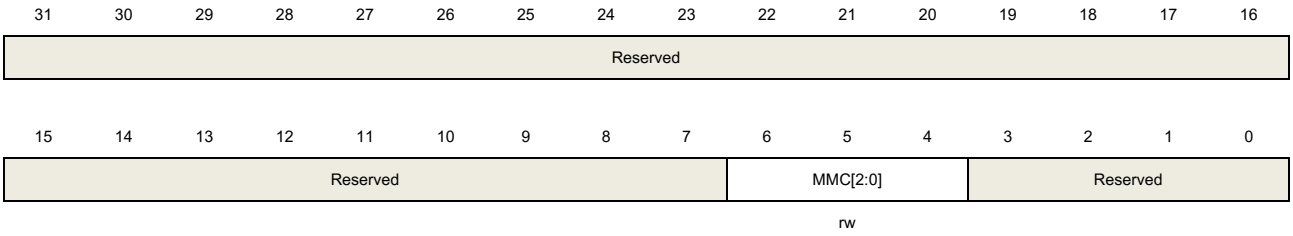
		This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. **Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
0	CEN	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and decoder mode.

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
31:7	Reserved	Must be kept at reset value.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.

		011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output. 100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE. 101: Reserved 110: Reserved 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														CH0IE	UPIE
														rw	rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CH0OF	Reserved.							CH0IF	UPIF
						rc_w0								rc_w0	rc_w0

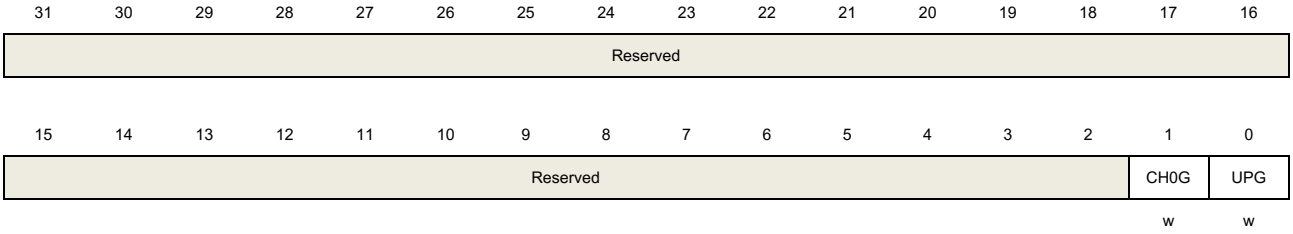
Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:2	Reserved	Must be kept at reset value.
1	CH0IF	Channel 0 's capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 0: No Channel 1 interrupt occurred 1: Channel 1 interrupt occurred
0	UPIF	Update interrupt flag This bit is set by hardware on an update event 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).



Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already

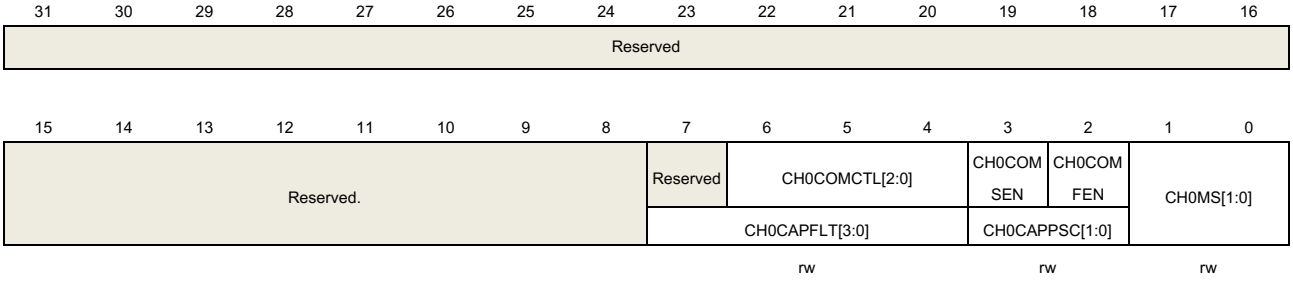
		high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time. 0: No generate an update event 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
31:7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP bits. 000: Timing mode. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced to low level. 101: Force high. O0CPRE is forced to high level. 110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise. 111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high

when the counter is larger than TIMERx_CH0CV, and low otherwise.
If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from "Timing" mode to "PWM" mode or the comparison result changes.

3	CH0COMSEN	Channel 0 compare output shadow enable When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled. 0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)
2	CH0COMFEN	Channel 0 output compare fast enable 0: Channel 0 output quickly compare disable.Only the valid edge of the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the result of the comparison to activate CH0_O output is 5 clock cycles. 1: Channel 0 output quickly compare enable.Both the valid edge of the trigger input signal and the comparison result will generate a comparison match, and CH0_O is set to the compare level.The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.
1:0	CH0MS[1:0]	Channel 0 I/O mode selection 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. (CH0EN bit in TIMERx_CHCTL2 register is reset).). 00: Channel 0 is configured as output 01: Channel 0 is configured as input, IS0 is connected to CI0FE0 1x: Reserved

Input capture mode:

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:4	CH0CAPFLT[3:0]	Channel 0 input capture filter control The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to f_{SAMP} and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:

CH0CAPFLT [3:0]	Times	f_{SAMP}
4'b0000	Filter disabled.	
4'b0001	2	f_{CK_TIMER}

4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	8	

3:2 CH0CAPPSC[1:0] Channel 0 input capture prescaler

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 clear.

00: Prescaler disable, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on every 8 channel input edges

1:0 CH0MS[1:0] Channel 0 mode selection

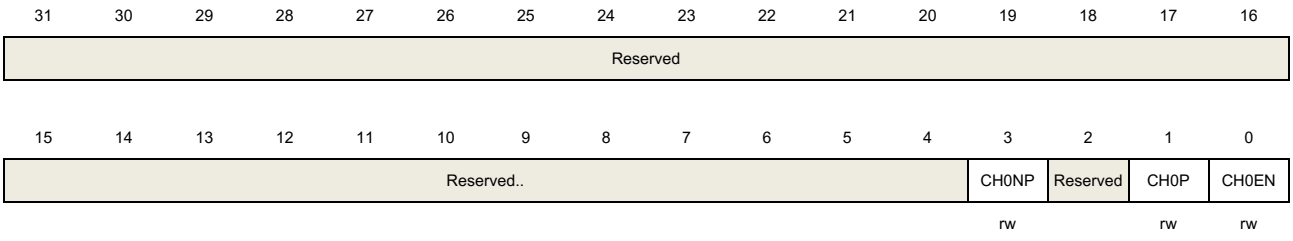
Same as output compare mode

Channel control register 2 (TIMERx_CHCTL2)

Address offset: 0x20

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	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 0 complementary output high level is active level

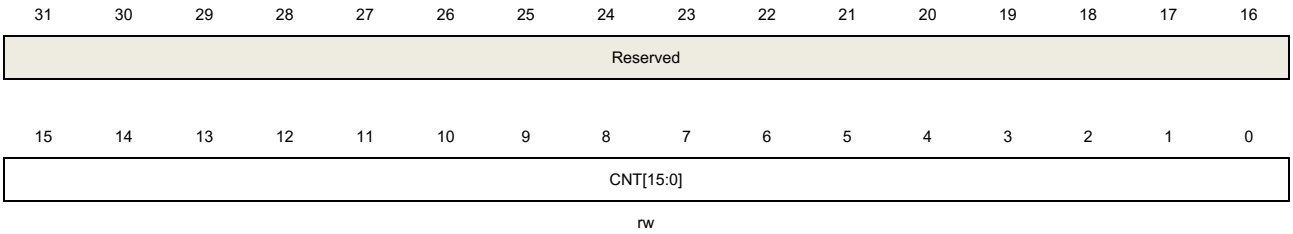
		1: Channel 0 complementary output low level is active level When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.
2	Reserved	Must be kept at reset value.
1	CH0P	Channel 0 capture/compare polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted.
0	CH0EN	Channel 0 capture/compare function enable When channel 0 is configured in input mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in output mode, setting this bit enables the capture event in channel0. 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
31:16	Reserved	Must be kept at reset value.
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed 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 The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed 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 This bit-filed 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).

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 or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

Input remap register (TIMERx_IRMP) (x=10)

Address offset: 0x50

Reset value: 0x0000 0000

This register 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														CI0_RMP	
														rw	

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1:0	CI0_RMP	Channel 0 input remap 00, 01, 11: GPIO pin. Refer to GPIO remap table 10: HXTAL_DIV(Clock used for RTC which is HXTAL clock divided by RTCDIV bits in RCU_CFG0 register

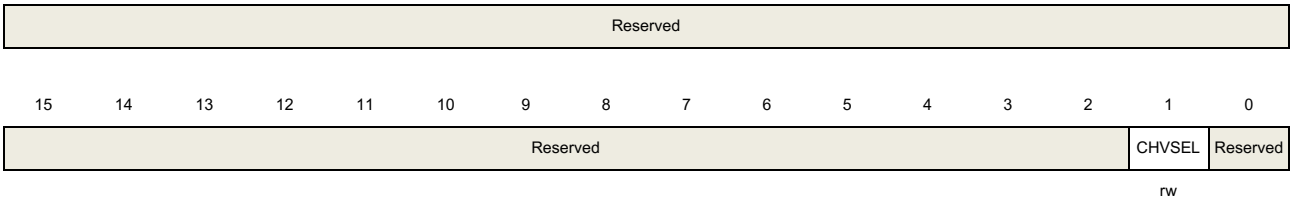
Configuration register (TIMERx_CFG)

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



Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

22.5. Basic timer (TIMERx, x=5, 6)

22.5.1. Overview

The basic timer module (Timer5, 6) reference is a 16-bit counter that can be used as an unsigned counter. The basic timer can be configured to generate DMA request and TRGO to DAC.

22.5.2. Characteristics

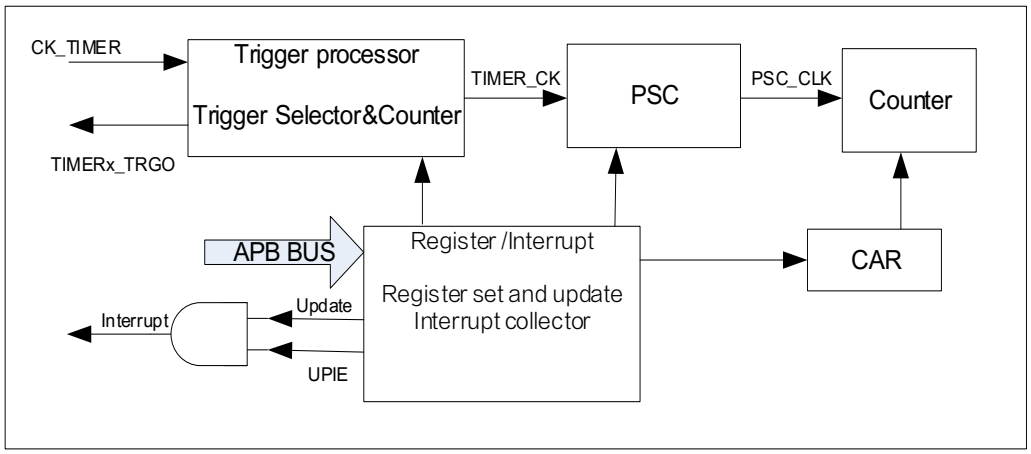
- Counter width: 16bit.
- Source of count clock is internal clock only.
- Multiple counter modes: count up.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Auto-reload function.
- Interrupt output or DMA request on update event.

22.5.3. Function overview

Block diagram

Figure 22-69. Basic timer block diagram provides details on the internal configuration of the basic timer.

Figure 22-69. Basic timer block diagram



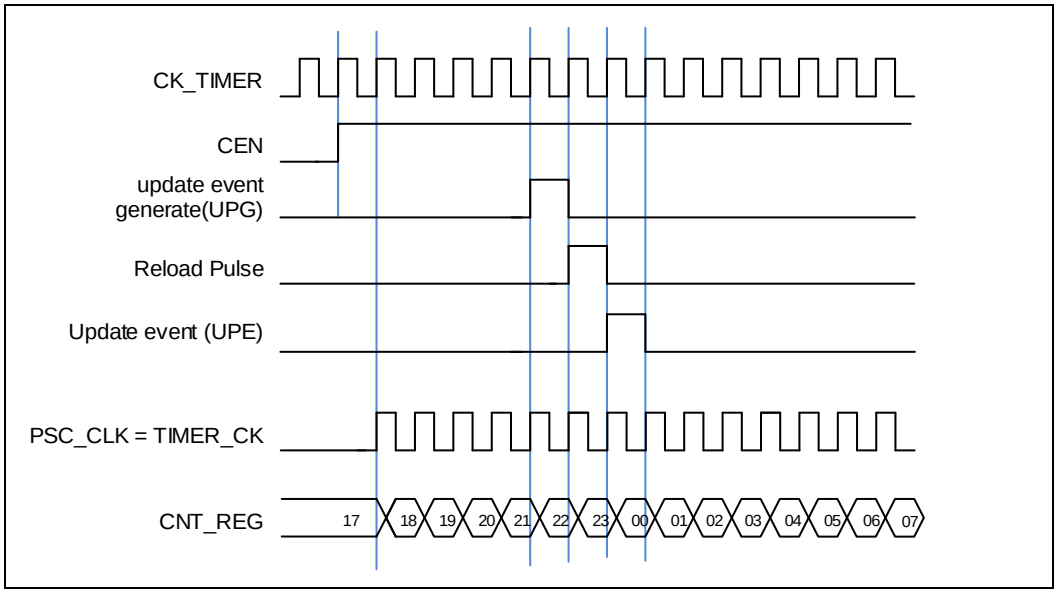
Clock source configuration

The basic TIMER can only being 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.

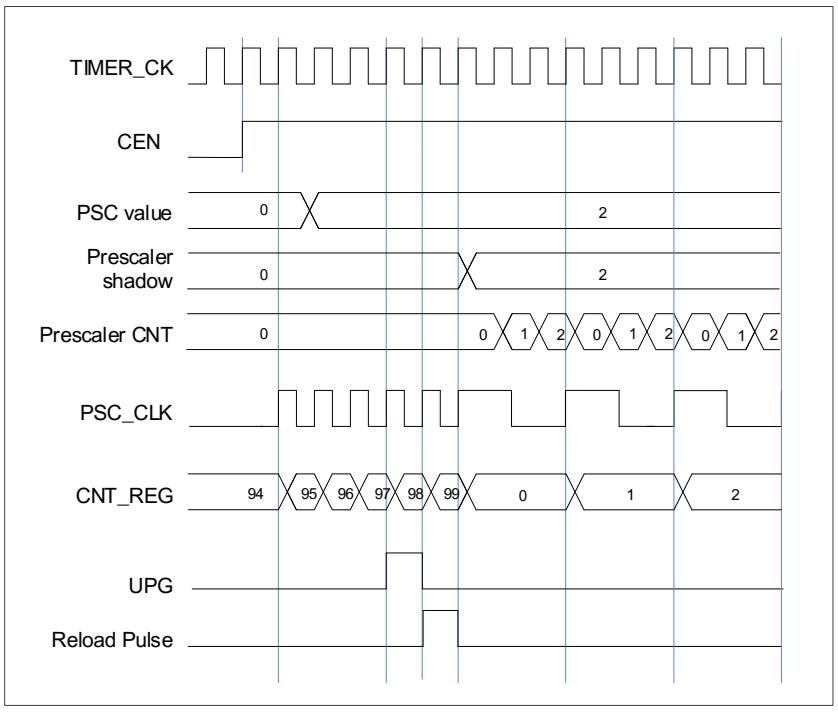
Figure 22-70. Timing chart of internal clock divided by 1



Clock prescaler

The counter clock (PSC_CK) is obtained by the TIMER_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx_PSC). The new written prescaler value will not take effect until the next update event.

Figure 22-71. Timing chart of PSC value change from 0 to 2



Counter up counting

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 will start counting up from 0 again. The update event is generated at each counter overflow.

When the update event is set by 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 shadow registers (counter 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.

Figure 22-72. Timing chart of up counting mode, PSC=0/2

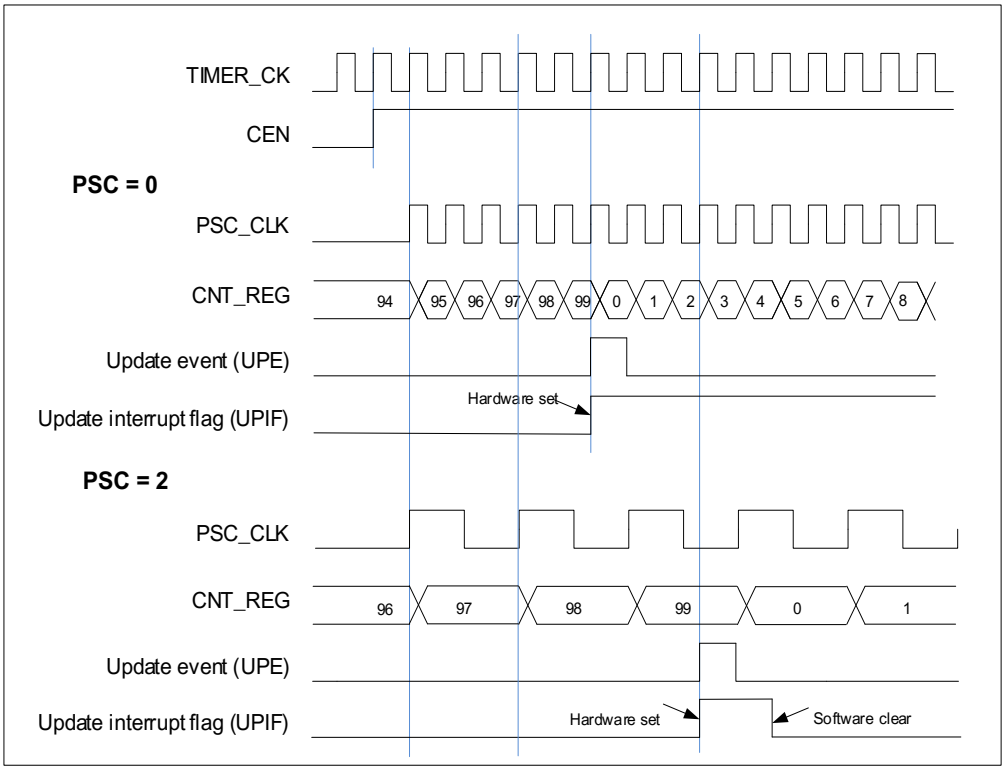
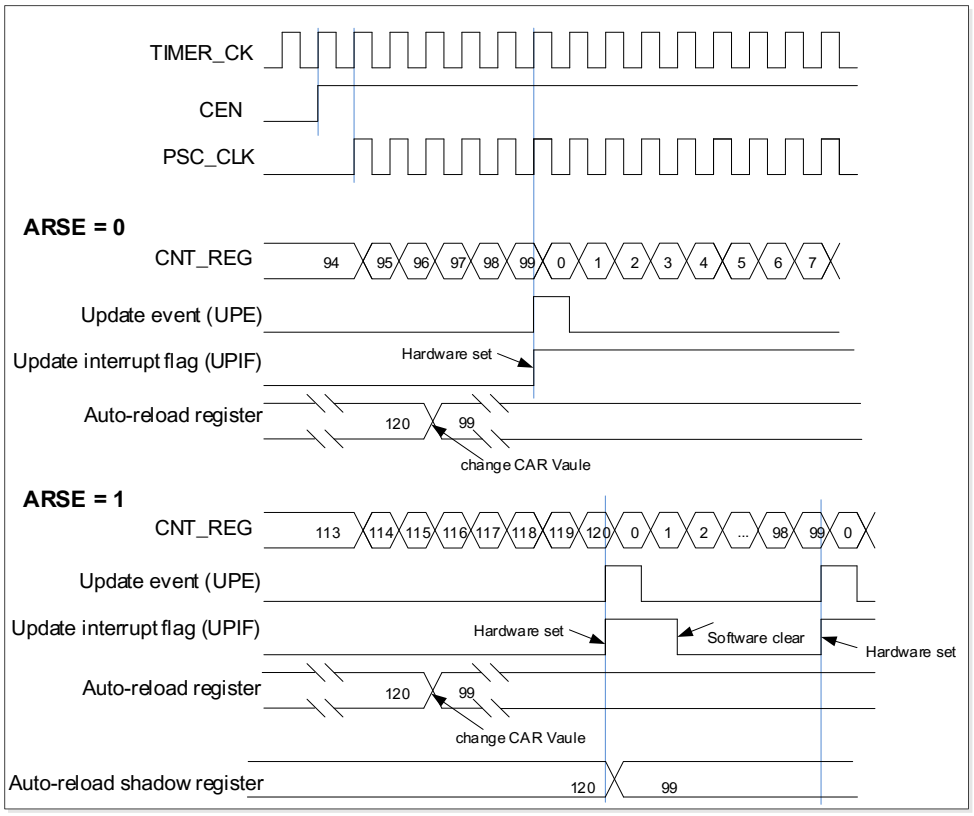


Figure 22-73. Up-counter timechart, change TIMERx_CAR on the go



Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx_CTL0. When you set SPM, the counter will be clear and stop when the next update event.

Once the timer is set to operate in the single pulse mode, it is necessary to set the timer enable bit CEN in the TIMERx_CTL0 register to 1 to enable the counter, then the CEN bit keeps 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 using software, the counter will be stopped and its value held.

Timer debug mode

When the Cortex®-M33 halted, and the TIMERx_HOLD configuration bit in DBG_CTL2 register set to 1, the TIMERx counter stops.

22.5.4. Register definition

TIMER5 base address: 0x4000 1000

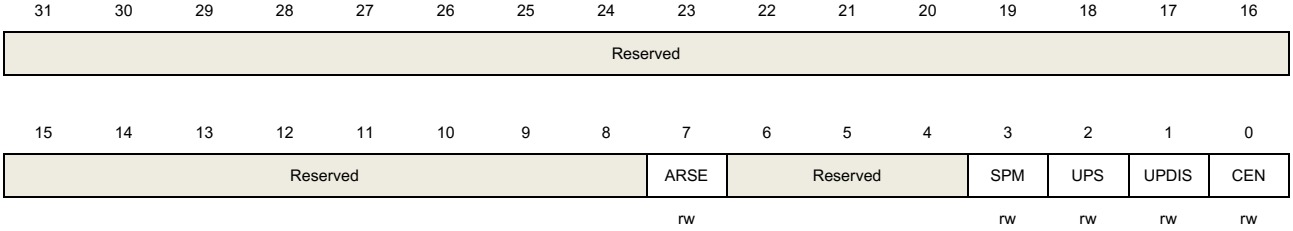
TIMER6 base address: 0x4000 1400

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
31:8	Reserved	Must be kept at reset value.
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:4	Reserved	Must be kept at reset value.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event

The restart mode generates an update event.

1: Update event disable.

Note: When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.

0 CEN Counter enable

0: Counter disable

1: Counter enable

The CEN bit must be set by software when timer works in external clock, pause mode and decoder mode.

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									MMC[2:0]			Reserved			
									rw						

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: When a counter reset event occurs, a TRGO trigger signal is output. The counter resert source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a conter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high 010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.
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 0: disabled 1: enabled
7:1	Reserved	Must be kept at reset value.
0	UPIE	Update interrupt enable 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															

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 This bit is set by hardware on an update event 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															

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 cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time. 0: No generate an update event 1: Generate an update event

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
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]	This bit-filed indicates the current counter value. Writing to this bit-filed 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															

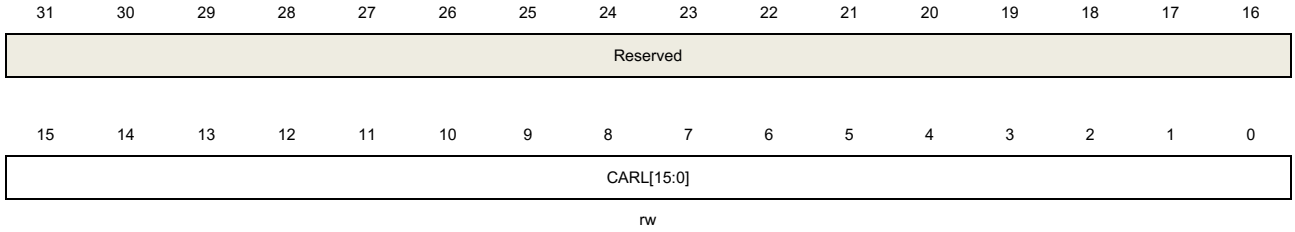
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	PSC[15:0]	Prescaler value of the counter clock The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

Counter auto reload register (TIMERx_CAR)

Address offset: 0x2C

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	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

23. Universal synchronous/asynchronous receiver /transmitter (USART)

23.1. 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 (PCLK1 or PCLK2) to produce a dedicated baud rate lock 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 and half-duplex 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 data bits and the TX/RX pins can be configured independently and flexibly.

ALL USARTs support DMA function for high-speed data communication.

23.2. Characteristics

- NRZ standard format
- Asynchronous, full duplex communication
- Half duplex single wire communications
- Programmable baud-rate generator
  - Divided from the peripheral clocks, PCLK2 for USART0/5, PCLK1 for USART1/2 and UART3/4/6/7.
  - Oversampling by 8 or 16
  - Maximum speed up to 12.5 MBits/s (PCLK2 100M and oversampling by 8)
- Fully programmable serial interface characteristics:
  - Even, odd or no-parity bit generation/detection
  - A data word length can be 8 or 9 bits
  - 0.5, 1, 1.5 or 2 stop bit generation
- Transmitter and receiver can be enabled separately
- Hardware flow control protocol (CTS/RTS)
- DMA request for data buffer access
- 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 frame or address match detection
- Various status flags:
  - Flags for transfer detection: receive buffer not empty (RBNE), transmit buffer empty (TBE), transfer complete (TC), and busy (BSY).
  - 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)
  - Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF)
  - Interrupt occurs at these events when the corresponding interrupt enable bits are set

While USART0/1/2/5 is fully implemented, UART3/4/6/7 is only partially implemented with the following features not supported.

- Smartcard mode
- Synchronous mode
- Hardware flow control protocol (CTS/RTS)
- Configurable data polarity

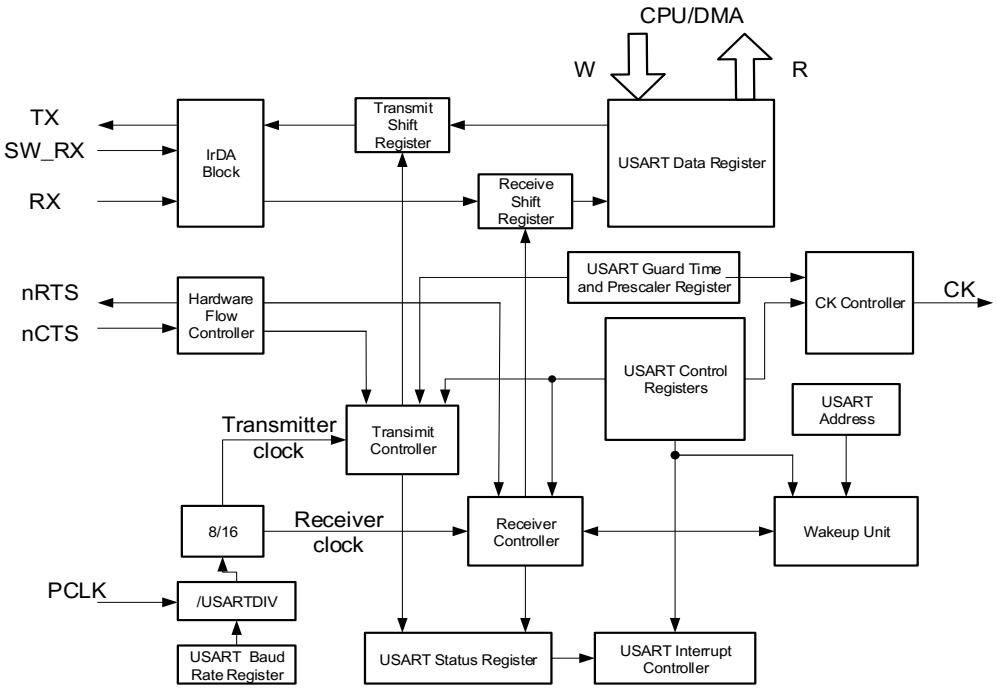
23.3. Function overview

The interface is externally connected to another device by the main pins listed in ***Table 23-1. Description of USART important pins***.

Table 23-1. 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

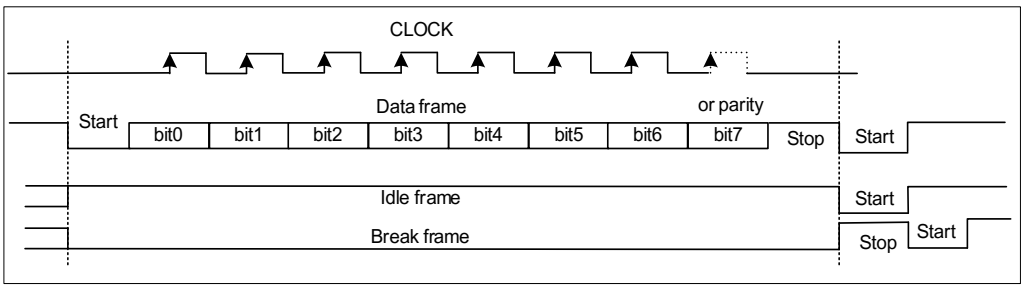
Figure 23-1. USART module block diagram



23.3.1. USART frame format

The USART frame starts with a start bit and end up with a number of stop bits. The length of the data frame is configured by the WL bit in the USART_CTL0 register. The last data bit can be used as parity check bit by setting the PCEN bit in USART_CTL0 register. When the WL bit is reset, the parity bit is the 7th bit. When the WL bit is set, the parity bit is the 8th bit. The method of calculating the parity bit is selected by the PM bit in USART_CTL0 register.

Figure 23-2. 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 23-2. 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

STB[1:0]	stop bit length (bit)	usage description
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.

23.3.2. 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 peripheral clock:

In case of oversampling by 16, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}} \quad (23\text{-}1)$$

In case of oversampling by 8, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{8 \times \text{Baud Rate}} \quad (23\text{-}2)$$

For example, when oversampled by 16:

1. Get USARTDIV by caculating the value of USART_BUAD:
   If USART_BUAD=0x21D, then INTDIV=33 (0x21), FRADIV=13 (0xD).
   USARTDIV=33+13/16=33.81.
2. Get the value of USART_BUAD 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_BUAD=0x1E6.

   Note: If the roundness of FRADIV is 16 (overflow), the carry must be added to the integer part.

23.3.3. USART transmitter

If the transmit enable bit (TEN) in USART_CTL0 register is set, when the transmit data buffer is not empty, the transmitter shift 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_CTL3 register. Clock pulses can be 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.

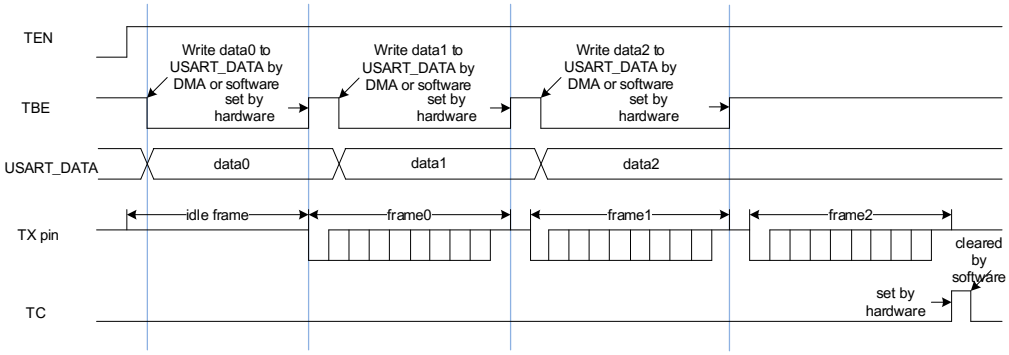
After power on, the TBE bit is high by default. Data can be written to the USART_DATA when the TBE bit of the USART_STAT0 register is asserted. The TBE bit is cleared by writing USART_DATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written to the USART_DATA 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_DATA 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_STAT0 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 23-3. USART transmit procedure***. The software can follow this flow:

1. Set the UEN bit in USART_CTL0 to enable the USART.
2. Write the WL bit in USART_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART_CTL1 to configure the number of stop bits.
4. Enable DMA (DENT bit) in USART_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART_BAUD.
6. Set the TEN bit in USART_CTL0.
7. Wait for the TBE being asserted.
8. Write the data to in the USART_DATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

Figure 23-3. 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. This bit can be cleared by a software sequence: reading the USART_STAT0 register and then writing the USART_DATA register. If the multibuffer communication is selected (DENT=1), this bit can also be cleared by writing 0 directly.

23.3.4. USART receiver

After power on, the USART receiver can be enabled by the follow procedure:

1. Set the UEN bit in USART_CTL0 to enable the USART.
2. Write the WL bit in USART_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART_CTL1.
4. Enable DMA (DENR bit) in USART_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART_BAUD.
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.

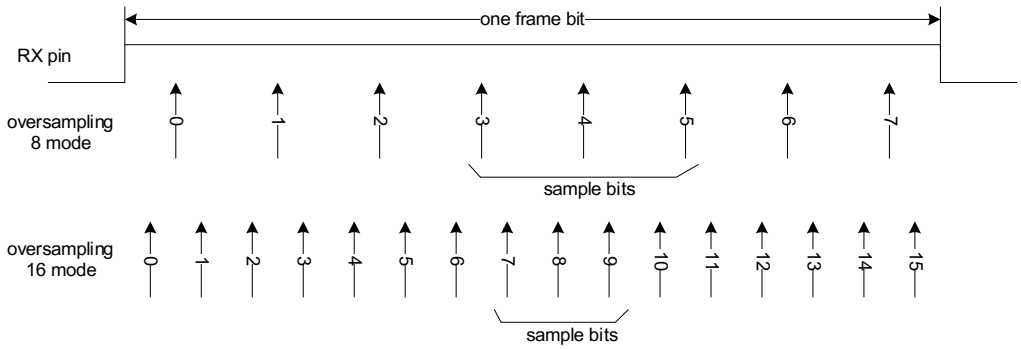
When a frame is received, the RBNE bit in USART_STAT0 is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART_CTL0 register. The status of the reception are stored in the USART_STAT0 register.

The software can get the received data by reading the USART_DATA register directly, or through DMA. The RBNE bit is cleared by a read operation on the USART_DATA 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 there are two or more samples of a frame bit is 0, the frame bit is confirmed as a 0, else 1. If the three samples of any bit of a frame 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 is generated, If the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set. If the OSB bit in USART_CTL2 register is set, the receiver get only one sample to evaluate a bit value. In this situation, no noisy error will be detected.

Figure 23-4. Receiving a frame bit by oversampling method (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_STAT0 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_STAT0 register will be set. An interrupt will be generated if the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set.

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_STAT0 register will be set. An interrupt is generated, if the receive DMA is enabled and the ERRIE bit in USART_CTL2 register is set, or if the RBNEIE is set.

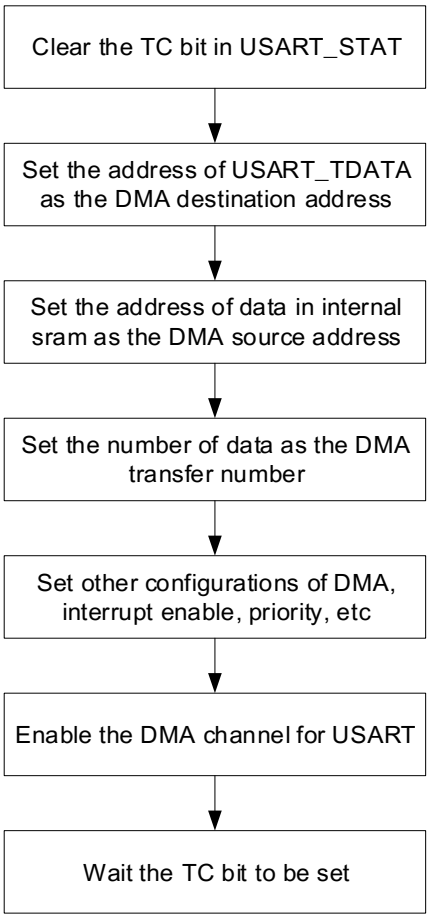
If a noise error (NERR), parity error (PERR), frame error (FERR) or overrun error (ORERR) is generated during a receiving process, then NERR, PERR, FERR or ORERR will be set at same time with RBNE. If DMA is disabled, the software needs to check whether the RBNE interrupt is caused by noise error, parity error, framing error or overflow error when the RBNE interrupt occurs.

23.3.5. 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 23-5. Configuration step when use DMA for USART transmission***

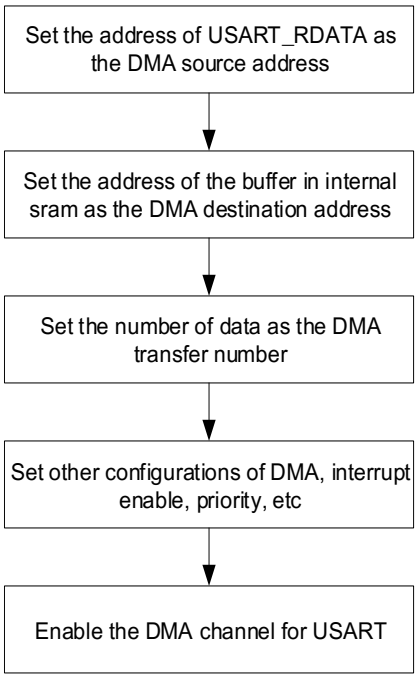
Figure 23-5. Configuration step when use DMA for USART transmission



After all of the data frames are transmitted, the TC bit in USART_STAT0 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 step is shown in ***Figure 23-6. Configuration step when use 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_STAT0.

Figure 23-6. Configuration step when use DMA for USART reception

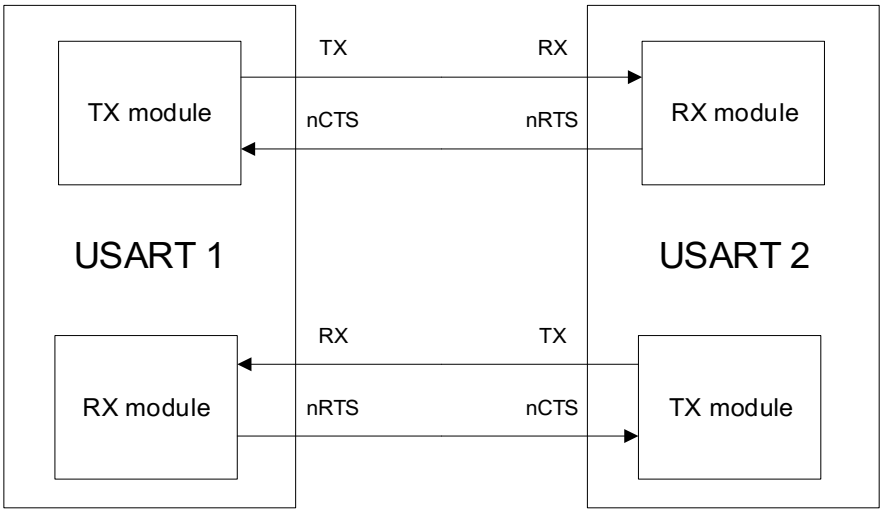


When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt can be generated in the DMA module.

23.3.6. 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 23-7. Hardware flow control between two USARTs



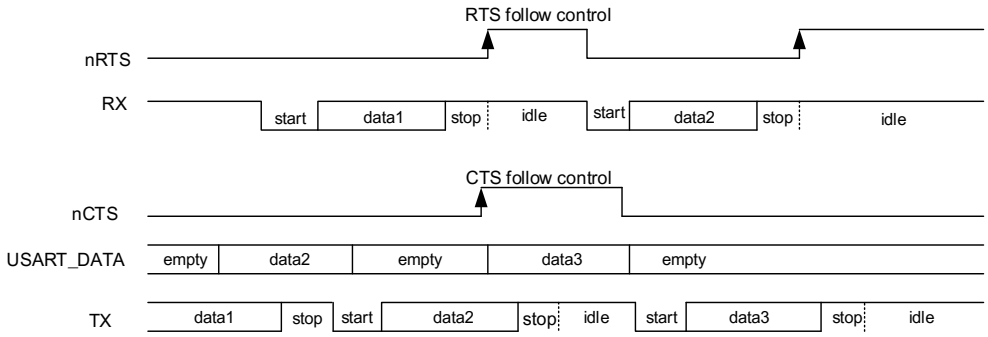
RTS flow control

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, and can be cleared by reading the USART_DATA register.

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_STAT0 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 23-8. Hardware flow control



If the CTS flow control is enabled, the CTSF bit in USART_STAT0 is set when the nCTS pin toggles. An interrupt is generated if the CTSIE bit in USART_CTL2 is set.

23.3.7. 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 setting the RWU bit in USART_CTL0 register.

If a USART is in mute mode, all of the receive status bits cannot be set. Software can wake up the USART by resetting the RWU bit.

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. When 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_STAT0 will not be set.

When the WM bit 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 treaded as a data frame. If the LSB 4 bits of an address

frame are the same as the ADDR[3:0] bits in the USART_CTL1 register, the hardware will clear the RWU bit and exit the mute mode. The RBNE bit will be set for the frame that wakes up the USART. The status bits are available in the USART_STAT0 register. If the LSB 4 bits of an address frame differ from the ADDR[3:0] bits in the USART_CTL1 register, the hardware sets the RWU bit and enters mute mode automatically. In this situation, the RBNE bit is not set.

If the address match method is selected, IDLEF can not be set and the receiver does not check the parity value of an address frame by default. If the PCM bit in USART_CHC and PCEN bit in USART_CTL0 are set, the MSB bit will be checked as the parity bit, and the bit preceding the MSB bit is detected as the address flag.

The RWU bit can be written to as 0 or 1 when the RBNE is 0.

23.3.8. LIN mode

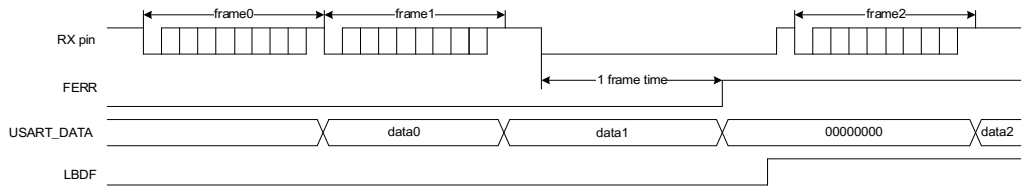
The local interconnection network mode is enabled by setting the LMEN bit in USART_CTL1. The CKEN, WL, STB[1:0] bits in USART_CTL1 and the SCEN, HDEN, IREN bits in USART_CTL2 should be reset in LIN mode.

When transmit a normal data frame, the transmission procedure is the same as the normal USART mode. When the SBKCMD bit in USART_CTL0 is set, the USART transmit continuous 13 '0' bits, followed by 1 stop bit.

The break detection function is totally independent from 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 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 in USART_STAT0 is set. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.

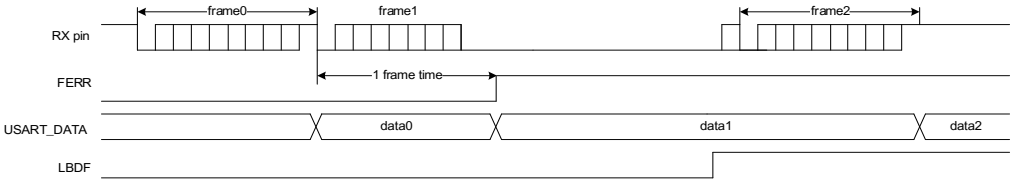
As shown in ***Figure 23-9. Break frame occurs during idle state***, if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all '0' frame, with an asserted FERR status.

Figure 23-9. Break frame occurs during idle state



As shown in ***Figure 23-10. 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 23-10. Break frame occurs during a frame



23.3.9. 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 he 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 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).

If the REN bit in USART_CTL0 is set, the receiver works differently from the normal USART reception method. The receiver samples the data on the capture edge of the CK pin without any oversampling.

Figure 23-11. Example of USART in synchronous mode

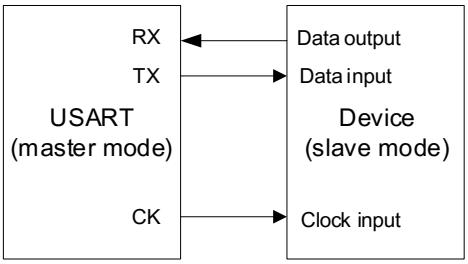
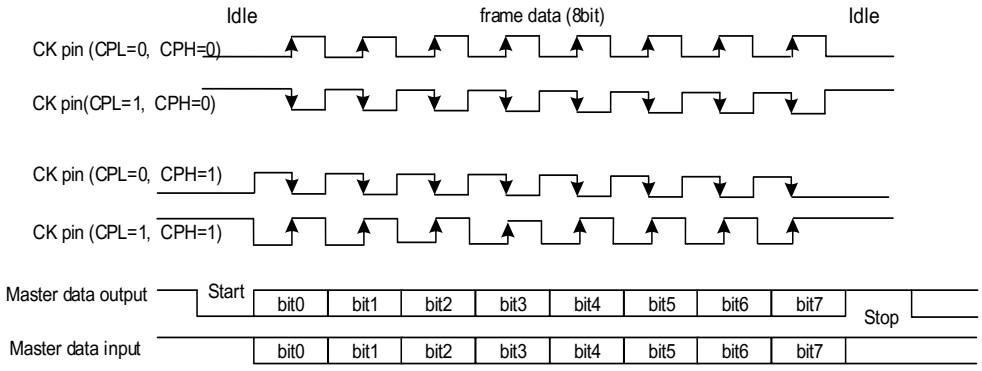


Figure 23-12. 8-bit format USART synchronous waveform (CLEN=1)

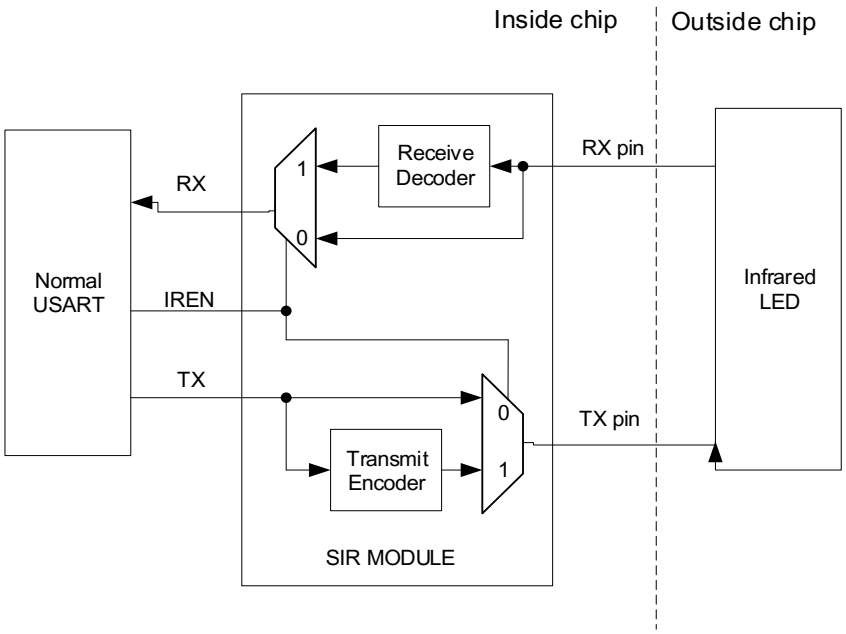


23.3.10. IrDA SIR ENDEC mode

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 put the demodulated data frame to the USART receiver. The baud rate should not be larger than 115200 for the encoder.

Figure 23-13. IrDA SIR ENDEC module

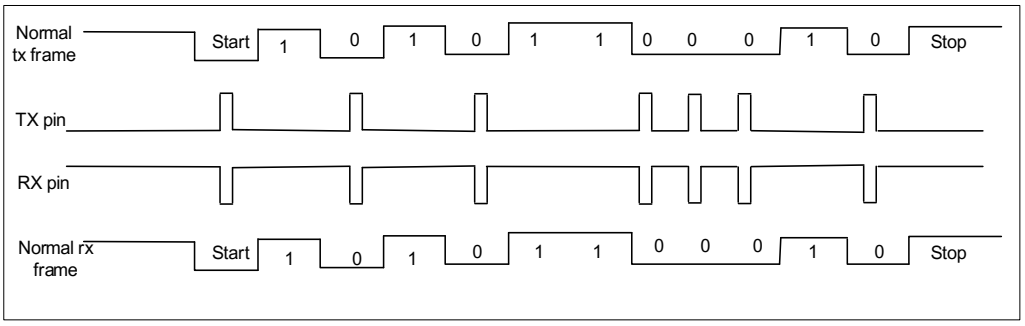


In IrDA mode, the polarity of the TX pin and RX pin is different. The TX pin is usually at low state, while the RX pin is usually at high state. The IrDA pins keep stable to represent the logic '1', while an infrared light pulse on the IrDA pins (a Return to Zero signal) represent the logic '0'. The 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 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 23-14. 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 divide 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.

23.3.11. 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.

In the half-duplex mode the receive line is internally connected to the TX pin, and the RX pin is no longer used. The TX pin should be configured as output open drain mode. The software should make sure that the transmission and reception process never conflict with each other.

23.3.12. Smartcard (ISO7816-3) mode

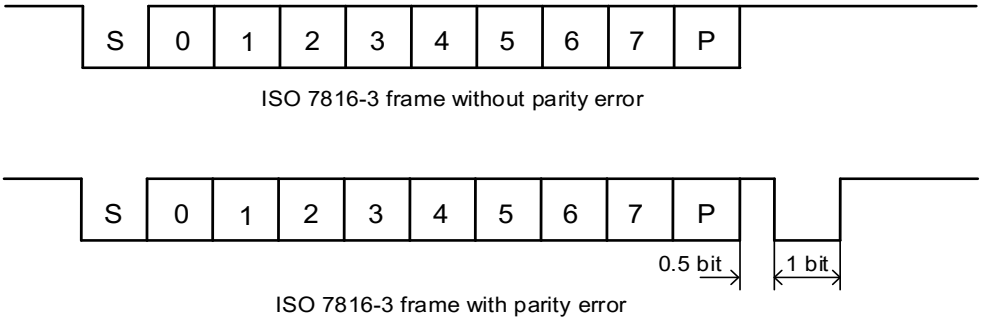
The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. Both the character (T=0) mode and the block (T=1) mode are supported. The smartcard mode is enabled by setting the SCEN bit in USART_CTL2. The LMEN bit in USART_CTL1 and HDEN, IREN bits in USART_CTL2 should be cleared in smartcard mode.

A clock is provided to the external smartcard through the CK pin after the CKEN bit is set. The clock is divided from the PCLK. The divide ratio is configured by the PSC[4:0] bits in USART_GP register. The CK pin only provides a clock source to the smartcard.

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 an external pull-up resistor will be needed, which drives a bidirectional line that is also driven by the smartcard. The data

frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits. The 0.5 stop bit may be configured for a receiver.

Figure 23-15. ISO7816-3 frame format



Character (T=0) mode

Comparing 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 count 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 2.5 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 framing 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 occurred 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 signals the error as a parity error if the received character is still erroneous after the maximum number of retries 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. This timeout period is expressed in baud time units. The RTF bit in USART_STAT1 will be asserted, if no answer is received from the card before the expiration of this period. An interrupt is generated if the RTIE bit in USART_CTL3 is set. The USART generates a RBNE interrupt if the first character is received before the expiration of the RT[23:0] period. If DMA is used to read from the smartcard in block mode, the DMA must be enabled only after the first character is received.

After the first character is received, the RT[23:0] bits should be configured to the CWT (character wait time) - 11 to enable the automatic check of the maximum interframe gap between two consecutive characters. The RTF bit in USART_STAT1 will be asserted, if the smartcard stop sending characters for the RT[23:0] period.

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). The block length counter counts up from 0 to the maximum value of BL[7:0]+4. The end of the block status (EBF bit in USART_STAT1) is set after the block length counter reaches the maximum value. An interrupt is generated if the EBIE bit in USART_CTL3 is set. The RTF bit may be set in case that an error in the block length.

If DMA is used for reception, this register field must be programmed to the minimum value (0x0) before the start of the block. With this value, the end of the block interrupt occurs after the 4th received character. The block length value can be read from the receive buffer at the third byte.

If DMA is not used for reception, the BL[7:0] bits should be firstly configured with the maximum value 0xFF to avoid generating an EBF status. The real block length value can be reconfigured to the BL[7:0] bits after the third byte is received.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

When the direct convention is selected, the LSB of the data frame is transferred first, high state on the TX pin represents logic '1', the parity check mode is even. In this case the MSBF and DINV bits in USART_CTL3 should be reset.

When the inverse convention is selected, the MSB of the data frame is transferred first, high state on the TX pin represents logic '0', the parity check mode is even. In this case the MSBF and DINV bits in USART_CTL3 should be set.

23.3.13. USART interrupts

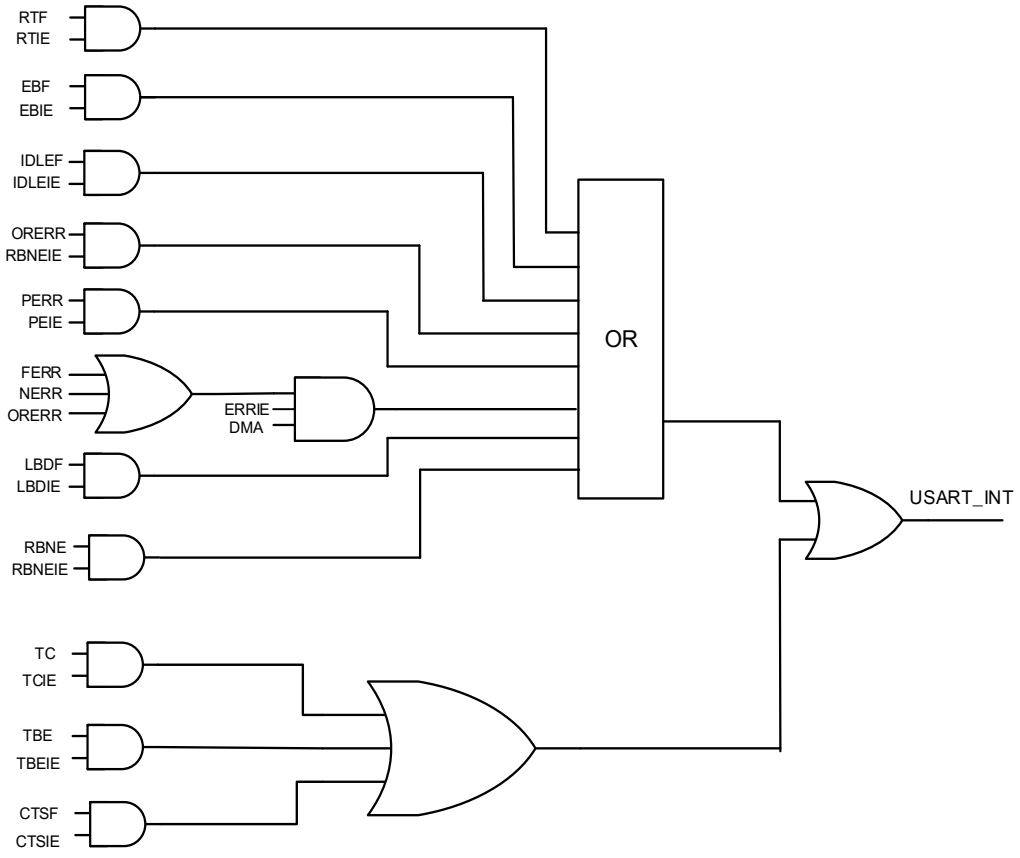
The USART interrupt events and flags are listed in *Table 23-3. USART interrupt requests*.

Table 23-3. USART interrupt requests

Interrupt event	Event flag	Enable Control bit
Transmit data buffer empty	TBE	TBEIE
CTS toggled flag	CTSF	CTSIE
Transmission complete	TC	TCIE
Received buff not empty	RBNE	RBNEIE
Overrun error	ORERR	
Idle frame	IDLEF	IDLEIE
Parity error	PERR	PERRIE
Break detected flag in LIN mode	LBDF	LBDIE
Receiver timeout	RTF	RTIE
End of block	EBF	EBIE
Reception errors (noise flag, overrun error, framing error) in DMA reception	NERR or ORERR or FERR	ERRIE

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 23-16. USART interrupt mapping diagram



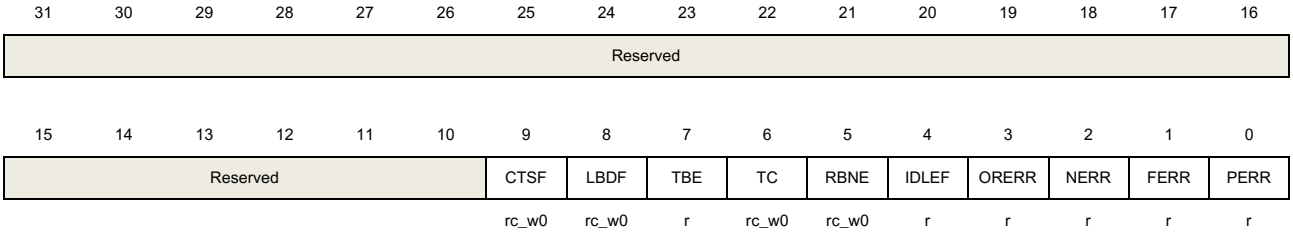
23.4. Register definition

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

23.4.1. Status register 0 (USART_STAT0)

Address offset: 0x00
Reset value: 0x0000 00C0

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



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	CTSF	CTS change flag If CTSEN bit in USART_CTL2 is set, this bit set by hardware when the nCTS input toggles. An interrupt occurs if the CTSIE bit in USART_CTL2 is set. Software can clear this bit by writing 0 to it. 0: The status of the nCTS line does not change. 1: The status of the nCTS line has changed. This bit is not available for UART3/4/6/7.
8	LBDF	LIN break detected flag LMEN bit in USART_CTL1 is set when LIN break is detected. An interrupt occurs if the LBDIE bit in USART_CTL1 is set. Software can clear this bit by writing 0 to it. 0: The USART does not detect a LIN break. 1: The USART has detected a LIN break.
7	TBE	Transmit data buffer empty. This bit is set after power on or when the transmit data has been transferred to the transmit shift register. An interrupt occurs if the TBEIE bit in USART_CTL0 is set. This bit is cleared when the software write transmit data to the USART_DATA

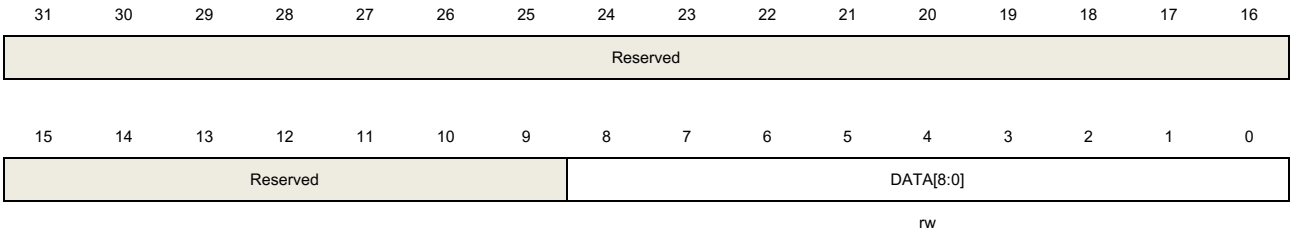
		register. 0: Transmit data buffer is not empty. 1: Transmit data buffer is empty.
6	TC	Transmission complete. This bit is set after power on. If the TBE bit has been set, this bit is set when the transmission of current data is complete. An interrupt occurs if the TCIE bit in USART_CTL0 is set. Software can clear this bit by writing 0 to it. 0: Transmission of current data is not complete. 1: Transmission of current data is complete.
5	RBNE	Read data buffer not empty. This bit is set when the read data buffer is filled with a data frame, which has been received through the receive shift register. An interrupt occurs if the RBNEIE bit in USART_CTL0 is set. Software can clear this bit by writing 0 to it or by reading the USART_DATA register. 0: Read data buffer is empty. 1: Read data buffer is not empty.
4	IDLEF	IDLE frame detected flag. This bit is set when the RX pin has been detected in idle state for a frame time. An interrupt occurs if the IDLEIE bit in USART_CTL0 is set. Software can clear this bit by reading the USART_STAT0 and USART_DATA registers one by one. 0: The USART module does not detect an IDLE frame. 1: The USART module has detected an IDLE frame.
3	ORERR	Overrun error This bit is set if the RBNE is not cleared and a new data frame is received through the receive shift register. An interrupt occurs if the ERRIE bit in USART_CTL2 is set. Software can clear this bit by reading the USART_STAT0 and USART_DATA registers one by one. 0: The USART does not detect a overrun error. 1: The USART has detected a overrun error.
2	NERR	Noise error flag When the OSB bit in USART_CTL2 is reset, this bit is set if the USART detects noise on the RX pin when receiving a frame. An interrupt occurs if the ERRIE bit in USART_CTL2 is set. Software can clear this bit by reading the USART_STAT0 and USART_DATA registers one by one. 0: The USART does not detect a noise error. 1: The USART has detected a noise error.
1	FERR	Frame error flag

		This bit is set when the RX pin is detected low during the stop bits of a receive frame. An interrupt occurs if the ERRIE bit in USART_CTL2 is set. Software can clear this bit by reading the USART_STAT0 and USART_DATA registers one by one. 0: The USART does not detect a framing error. 1: The USART has detected a framing error.
0	PERR	Parity error flag This bit is set when the parity bit of a receive frame does not match the expected parity value. An interrupt occurs if the PERRIE bit in USART_CTL0 is set. Software can clear this bit the sequence: read the USART_STAT0 register, and then read or write the USART_DATA register. 0: The USART does not detect a parity error. 1: The USART has detected a parity error.

23.4.2. Data register (USART_DATA)

Offset: 0x04
Reset value: Undefined

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



Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8:0	DATA[8:0]	Transmit or read data value Software can write these bits to update the transmit data or read these bits to get the receive data. If the parity check function is enabled, when transmit data is written to this register, the MSB bit (bit 7 or bit 8 depending on the WL bit in USART_CTL0) will be replaced by the parity bit.

23.4.3. Baud rate register (USART_BAUD)

Address offset: 0x08
Reset value: 0x0000 0000

The software must not write this register when the USART is enabled (UEN=1).
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
INTDIV [11:0]												FRADIV[3:0]			
rw												rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	INTDIV[11:0]	Integer part of baud-rate divider.
3:0	FRADIV[3:0]	Fraction part of baud-rate divider. Software must keep FRADIV[3] bit reset, if the oversample 8 mode is enabled.

23.4.4. Control register 0 (USART_CTL0)

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
OVSMOD	Reserved	UEN	WL	WM	PCEN	PM	PERRIE	TBEIE	TCIE	RBNEIE	IDLEIE	TEN	REN	RWU	SBKCMD
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	OVSMOD	Oversample mode 0: oversampling by 16 1: oversampling by 8 If SCEN=1, IREN=1 or LMEN=1, OVSMOD is forced to '0 by hardware.
14	Reserved	Must be kept at reset value.
13	UEN	USART enable 0: Disable USART. 1: Enable USART.
12	WL	Word length 0: 8 Data bits 1: 9 Data bits
11	WM	Wakeup method in mute mode 0: wake up by idle frame.

		1: wake up by address match.
10	PCEN	Parity check function enable. 0: Disable parity check function. 1: Enable parity check function.
9	PM	Parity mode 0: Even parity 1: Odd parity
8	PERRIE	Parity error interrupt enable. If this bit is set, an interrupt occurs when the PERR bit in USART_STAT0 is set. 0: Disable parity error interrupt. 1: Enable parity error interrupt.
7	TBEIE	Transmitter buffer empty interrupt enable. If this bit is set, an interrupt occurs when the TBE bit in USART_STAT0 is set. 0: Disable transmitter buffer empty interrupt. 1: Enable transmitter buffer empty interrupt.
6	TCIE	Transmission complete interrupt enable. If this bit is set, an interrupt occurs when the TC bit in USART_STAT0 is set. 0: Disable transmission complete interrupt. 1: Enable transmission complete interrupt.
5	RBNEIE	Read data buffer not empty interrupt and overrun error interrupt enable. If this bit is set, an interrupt occurs when the RBNE bit or the ORERR bit in USART_STAT0 are set. 0: Disable read data register not empty interrupt and overrun error interrupt. 1: Enable read data register not empty interrupt and overrun error interrupt.
4	IDLEIE	IDLE line detected interrupt enable. If this bit is set, an interrupt occurs when the IDLEF bit in USART_STAT0 is set. 0: Disable IDLE line detected interrupt. 1: Enable IDLE line detected interrupt.
3	TEN	Transmitter enable 0: Disable transmitter 1: Enable transmitter
2	REN	Receiver enable 0: Disable receiver 1: Enable receiver
1	RWU	Receiver wakeup from mute mode. Software can set this bit to make the USART work in mute mode and reset this bit to wake up the USART. In wake up by idle frame mode (WM=0), this bit can be reset by hardware when an idle frame has been detected. In wake up by address match mode (WM=1), this bit

		can be reset by hardware when receives an address match frame or set by hardware when receives an address mismatch frame. 0: Receiver in active mode. 1: Receiver in mute mode.
0	SBKCMD	Send break command Software can set this be to send a break frame. Hardware resets this bit automatically when the break frame has been transmitted. 0: Do not transmit a break frame 1: Transmit a break frame

23.4.5. Control register 1 (USART_CTL1)

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	LMEN	STB[1:0]		CKEN	CPL	CPH	CLEN	Reserved.	LBDIE	LBLEN	Reserved	ADDR[3:0]			
	rw	rw		rw	rw	rw	rw		rw	rw		rw			

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	LMEN	LIN mode enable 0: Disable LIN mode 1: Enable LIN mode
13:12	STB[1:0]	STOP bits length 00: 1 Stop bit 01: 0.5 Stop bit 10: 2 Stop bits 11: 1.5 Stop bit Only 1 stop bit and 2 stop bit are available for UART3/4/6/7.
11	CKEN	CK pin enable 0: Disable CK pin 1: Enable CK pin This bit reserved for UART3/4/6/7.
10	CPL	CK polarity This bit specifies the polarity of the CK pin in synchronous mode. 0: The CK pin is in low state when the USART is in idle state. 1: The CK pin is in high state when the USART is in idle state.

		This bit is reserved for UART3/4/6/7.
9	CPH	CK phase This bit specifies the phase of the CK pin in synchronous mode. 0: The capture edge of the LSB bit is the first edge of CK pin. 1: The capture edge of the LSB bit is the second edge of CK pin. This bit is reserved for UART3/4/6/7.
8	CLEN	CK Length This bit specifies the length of the CK signal in synchronous mode. 0: There are 7 CK pulses for an 8 bit frame and 8 CK pulses for a 9 bit frame. 1: There are 8 CK pulses for an 8 bit frame and 9 CK pulses for a 9 bit frame. This bit is reserved for UART3/4/6/7.
7	Reserved	Must be kept at reset value.
6	LBDIE	LIN break detected interrupt enable. If this bit is set, an interrupt occurs when the LBDF bit in USART_STAT0 is set. 0: Disable LIN break detected interrupt. 1: Enable LIN break detected interrupt.
5	LBLEN	LIN break frame length This bit specifies the length of a LIN break frame. 0: 10 bit 1: 11 bit
4	Reserved	Must be kept at reset value.
3:0	ADDR[3:0]	Address of the USART In wake up by address match mode (WM=1), the USART enters mute mode when the LSB 4 bits of a received frame do not equal the ADDR[3:0] bits, and wakes up when the LSB 4 bits of a received frame equal the ADDR[3:0] bits.

23.4.6. Control register 2 (USART_CTL2)

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				OSB	CTSIE	CTSEN	RTSEN	DENT	DENR	SCEN	NKEN	HDEN	IRLP	IREN	ERRIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions

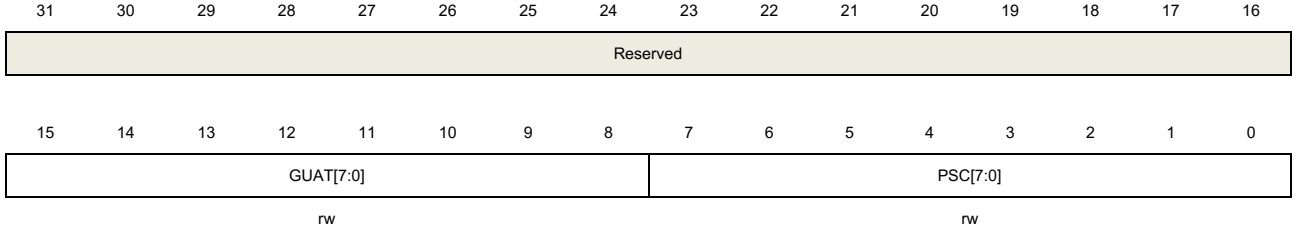
31:12	Reserved	Must be kept at reset value.
11	OSB	One sample bit method. This bit selects the sample method. When this bit is set, the USART get only one sample for a data bit instead of 3 samples per bit. The noise error flag (NERR) is disabled when the one sample bit method is selected. 0: Three sample bit method. 1: One sample bit method.
10	CTSIE	CTS interrupt enable. If this bit is set, an interrupt occurs when the CTSF bit in USART_STAT0 is set. 0: Disable CTS interrupt. 1: Enable CTS interrupt. This bit is reserved for UART3/4/6/7.
9	CTSEN	CTS enable This bit enables the CTS hardware flow control function. 0: Disable CTS hardware flow control. 1: Enable CTS hardware flow control. This bit is reserved for UART3/4/6/7.
8	RTSEN	RTS enable This bit enables the RTS hardware flow control function. 0: Disable RTS hardware flow control. 1: Enable RTS hardware flow control. This bit is reserved for UART3/4/6/7.
7	DENT	DMA request enable for transmission. 0: DMA request is disabled for transmission. 1: DMA request is enabled for transmission.
6	DENR	DMA request enable for reception. 0: DMA request is disabled for reception. 1: DMA request is enabled for reception.
5	SCEN	Smartcard mode enable This bit enables the smartcard work mode. 0: Disable smartcard mode 1: Enable smartcard mode This bit is reserved for UART3/4/6/7.
4	NKEN	NACK enable in Smartcard mode This bit enables the NACK transmission when parity error occurs in smartcard mode. 0: Disable NACK transmission 1: Enable NACK transmission This bit is reserved for UART3/4/6/7.
3	HDEN	Half-duplex enable

		This bit enables the half-duplex USART mode. 0: Disable Half duplex mode 1: Enable half duplex mode
2	IRLP	IrDA low-power This bit selects low-power mode of IrDA mode. 0: Normal mode 1: Low-power mode
1	IREN	IrDA mode enable This bit enables the IrDA mode of USART. 0: Disable IrDA 1: Enable IrDA
0	ERRIE	Error interrupt enable When DMA request for reception is enabled (DENR=1), if this bit is set, an interrupt occurs when any one of the FERR, ORERR and NERR bits in USART_STAT0 is set. 0: Disable error interrupt. 1: Enable error interrupt.

23.4.7. Guard time and prescaler register (USART_GP)

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.
15:8	GUAT[7:0]	Guard time value in Smartcard mode. TC flag assertion time is delayed by GUAT[7:0] baud clock cycles. These bits are not available for UART3/4/6/7.
7:0	PSC[7:0]	When the USART IrDA low-power mode is enabled, these bits specify the division factor that is used to divide the peripheral clock (PCLK1/PCLK2) to generate the low-power frequency. 00000000: Reserved - never program this value 00000001: divides by 1 00000010: divides by 2

...

11111111: divides by 255

When the USART works in IrDA normal mode, these bits must be set to 00000001.

When the USART smartcard mode is enabled, the PSC [4:0] bits specify the division factor that is used to divide the peripheral clock (APB1/APB2) to generate the smartcard clock (CK). The actual division factor is twice as the PSC [4:0] value.

00000: Reserved - never program this value.

00001: divides by 2

00010: divides by 4

...

11111: divides by 62

The PSC [7:5] bits are reserved in smartcard mode.

23.4.8. Control register 3 (USART_CTL3)

Address offset: 0x80

Reset value: 0x0000 0000

This register is not available for UART3/4/6/7.

This register 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				MSBF	DINV	TINV	RINV	Reserved		EBIE	RTIE	SCRTNUM[2:0]			RTEN
				rw	rw	rw	rw			rw	rw	rw			rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	MSBF	Most significant bit first. This bit specifies the sequence of the data bits in transmit and receive. 0: Data is transmitted/received with the LSB first. 1: Data is transmitted/received with the MSB first. This bit field cannot be written when the USART is enabled (UEN=1).
10	DINV	Data bit level inversion. This bit specifies the polarity of the data bits in transmission and reception. 0: Data bit signal values are not inverted. 1: Data bit signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
9	TINV	TX pin level inversion This bit specifies the polarity of the TX pin. 0: TX pin signal values are not inverted. 1: TX pin signal values are inverted.

		This bit field cannot be written when the USART is enabled (UEN=1).
8	RINV	RX pin level inversion This bit specifies the polarity of the RX pin. 0: RX pin signal values are not inverted. 1: RX pin signal values are inverted. This bit field cannot be written when the USART is enabled (UEN=1).
7:6	Reserved	Must be kept at reset value.
5	EBIE	Interrupt enable bit of end of block event. If this bit is set, an interrupt occurs when the EBF bit in USART_STAT1 is set. 0: End of block interrupt is disabled. 1: End of block interrupt is enabled.
4	RTIE	Interrupt enable bit of receive timeout event. If this bit is set, an interrupt occurs when the RTF bit in USART_STAT1 is set. 0: Receive timeout interrupt is disabled. 1: Receive timeout interrupt is enabled.
3:1	SCRTNUM[2:0]	Smartcard auto-retry number In smartcard mode, these bits specify the number of retries in transmit and receive. In transmission mode, a frame can be retransmitted by SCRTNUM times. If the frame is NACKed by (SCRTNUM+1) times, the FERR is set. In reception mode, a frame reception can be tried by (SCRTNUM+1) times. If the parity bit mismatch event occurs (SCRTNUM+1) times for a frame, the RBNE and PERR bits are set. When these bits are configured as 0x0, there will be no automatic retransmission in transmit mode.
0	RTEN	Receiver timeout enable. This bit enables the receive timeout counter of the USART. 0: Receiver timeout function disabled. 1: Receiver timeout function enabled.

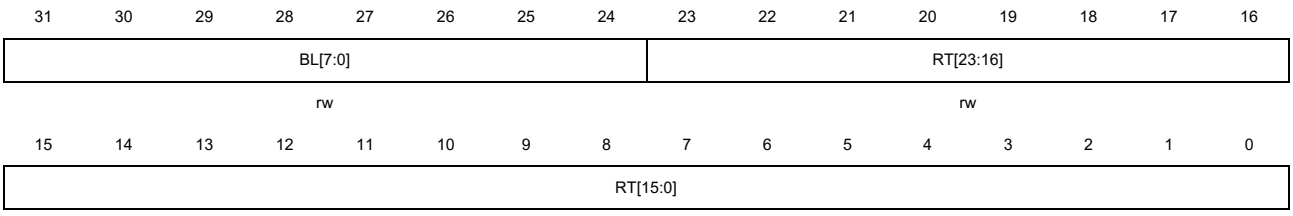
23.4.9. Receiver timeout register (USART_RT)

Address offset: 0x84

Reset value: 0x0000 0000

This register is not available for UART3/4/6/7.

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



rw

Bits	Fields	Descriptions
31:24	BL[7:0]	Block length These bits specify the block length in smartcard T=1 reception. Its value equals to the number of information characters + the length of the epilogue field (1-LEC/2-CRC) - 1. 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 in smartcard mode. In other modes, when REN=0 (receiver disabled), or when the EBF bit of USART_STAT1 is written to 0, the block length counter is reset.
23:0	RT[23:0]	Receiver timeout threshold. These bits are used to specify receiver timeout value in terms of number of baud clocks. If smartcard mode is not enabled, the RTF bit of USART_STAT1 is set if no new start bit is detected longer than RT bits time after the last received character. If smartcard mode is enabled, 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. 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 internal timeout counter. These bits must only be programmed once per received character. **Note:** When oversampling by 16, RT[23:0] cannot be configured as 0xFFFFFF.

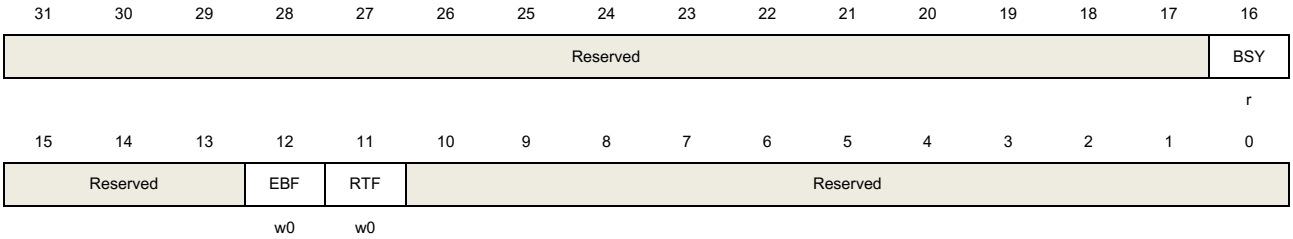
23.4.10. Status register 1 (USART_STAT1)

Address offset: 0x88

Reset value: 0x0000 00C0

This register is not available for UART3/4/6/7.

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



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	BSY	Busy flag

		This bit is set when the USART is receiving a data frame. 0: USART reception path is idle. 1: USART reception path is working.
15:13	Reserved	Must be kept at reset value.
12	EBF	End of block flag This bit is set when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4. An interrupt occurs if the EBIE bit in USART_CTL3 is set. Software can clear this bit by writing 0 to it. 0: End of block event not occurs. 1: End of block event has occurred.
11	RTF	Receiver timeout flag This bit is set when the RX pin is in idle state for longer than RT bits time. An interrupt occurs if the RTIE bit in USART_CTL3 is set. Software can clear this bit by writing 0 to it. 0: Receiver timeout event not occurs. 1: Receiver timeout event has occurred.
10:0	Reserved	Must be kept at reset value.

23.4.11. 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					BCM	PCM	HCM
							rc_w0						rw	rw	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. 0: No parity error is detected 1: Parity error is detected
7:3	Reserved	Must be kept at reset value.
2	BCM	Break frame coherence mode. 0: When the CTS flow is enabled, the transmitter does not check the nCTS input

		state to send a break. 1: When the CTS flow is enabled, the transmitter checks the nCTS input state to send a break. This bit is reserved for UART3/4/6/7.
1	PCM	Parity check coherence mode. 0: In case of wakeup by an address match: the MSB bit of the data is taken into account to identify an address but not the parity bit. And the receiver does not check the parity of the address data (PERR is not set in case of a parity error). 1: In case of wakeup by an address match, the receiver check the parity of the address data and PERR is set in case of a parity error.
0	HCM	Hardware flow control coherence mode. 0: nRTS signal equals to RBNE bit in USART_STAT0 register 1: nRTS signal is set when the last data bit (parity bit when PCE is set) has been sampled This bit is reserved for UART3/4/6/7.

24. Inter-integrated circuit interface (I2C)

24.1. Inter-integrated circuit interface (I2Cx, x=0, 1, 2)

24.1.1. 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 and fast-mode as well as CRC calculation and checking, SMBus (system management bus), PMBus (power management bus) and SAM_V (secure access and control module for validation) mode. It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

24.1.2. 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.
- Multi-master capability.
- Supports standard-mode (up to 100 kHz) and fast-mode (up to 400 kHz).
- Configurable SCL stretching in slave mode.
- Supports DMA mode.
- SMBus 2.0 and PMBus compatible.
- 2 Interrupts: one for successful byte transmission and the other for error event.
- Optional PEC (Packet Error Checking) generation and check.
- Supports SAM_V mode.
- Digital and analog noise filters.

24.1.3. Function overview

Figure 24-1. I2C module block diagram below provides details of the internal configuration of the I2C interface.

Figure 24-1. I2C module block diagram

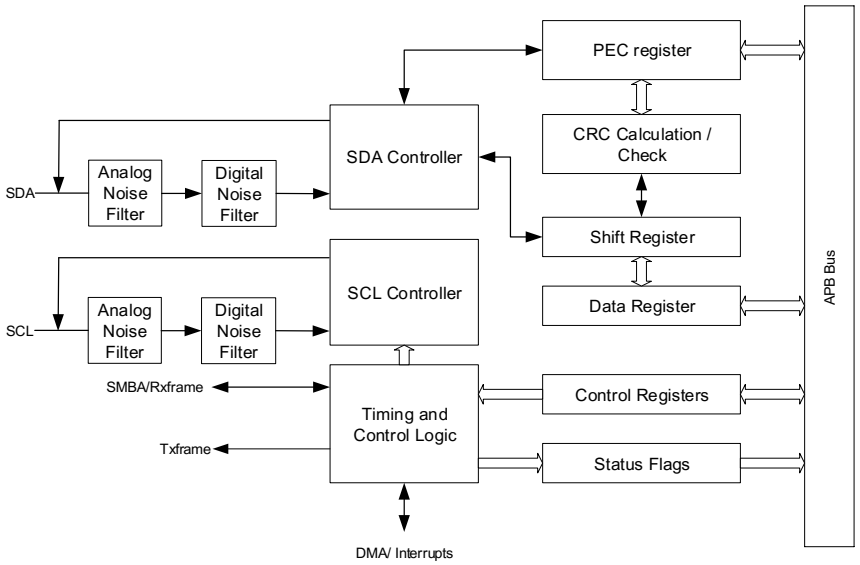


Table 24-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)

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
Synchronization	Procedure to synchronize the clock signals of two or more devices
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

SDA and SCL lines

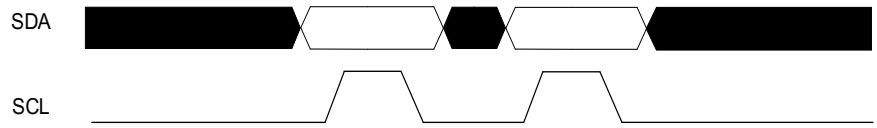
The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus.

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collect to perform the wired-AND function. Data on the I2C-bus can be transferred at rates of up to 100 Kbit/s in the standard mode and up to 400 Kbit/s in the fast mode. Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C-bus, the voltage levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of V_{DD}.

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 SDA line can only change when the clock signal on the SCL line is LOW (see ***Figure 24-2. Data validation***). One clock pulse is generated for each data bit to be transferred.

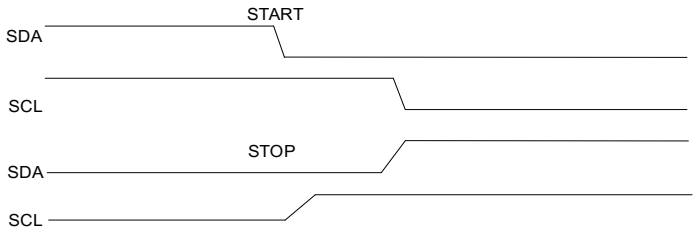
Figure 24-2. Data validation



START and STOP signal

All transmissions begin with a START and are terminated by a STOP (see ***Figure 24-3. START and STOP signal***). 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 24-3. START and STOP signal

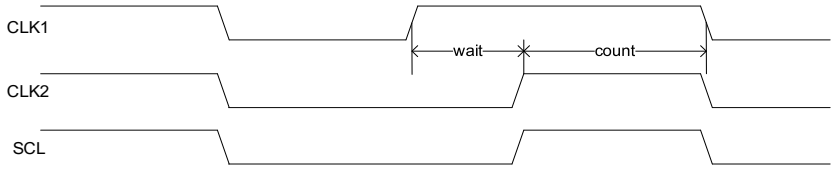


Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which master takes control of the bus and completes its transmission. This is done by clock synchronization and bus arbitration. In a single master system, clock synchronization and bus arbitration are unnecessary.

Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line causes the masters concerned to start counting their LOW period and, once a master clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see ***Figure 24-4. Clock synchronization***). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore held LOW by the master with the longest LOW period. Masters with shorter LOW period enter a HIGH wait-state during this time.

Figure 24-4. Clock synchronization



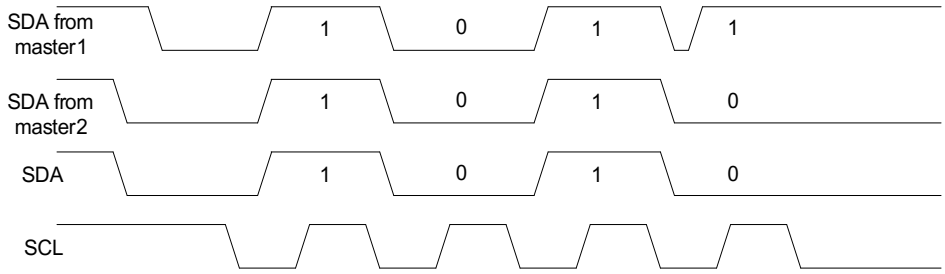
Arbitration

Arbitration, like synchronization, is part of the protocol where more than one master is used in the system. Slaves are not involved in the arbitration procedure.

A master may start a transfer only if the bus is free. Two masters may generate a START signal within the minimum hold time of the START signal which results in a valid START signal on the bus. Arbitration is then required to determine which master will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks whether the SDA level matches what it has been sent. This process may take many bits. Two masters can even complete an entire transmission without error, as long as the transmissions are identical. The first time a master tries to send a HIGH, but detects that the SDA level is LOW, then the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transmission.

Figure 24-5. SDA line arbitration



I2C communication flow

Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can be operated as either a transmitter or receiver, depending on the function of the device.

An I2C slave will continue to detect addresses after a START signal on I2C bus and compare the detected address with its slave address which is programmed by software. Once the two addresses match with each other, the I2C slave will send an ACK to the I2C bus and respond 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 responds to a General Call Address (0x00). The I2C block supports both 7-bit and 10-bit address modes.

An I2C master always initiates or ends a transfer using START or STOP signal and it's also responsible for SCL clock generation.

Figure 24-6. I2C communication flow with 7-bit address

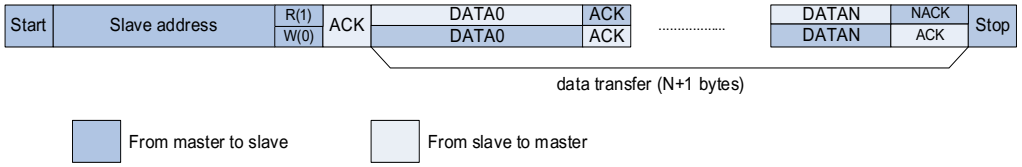


Figure 24-7. I2C communication flow with 10-bit address (Master Transmit)

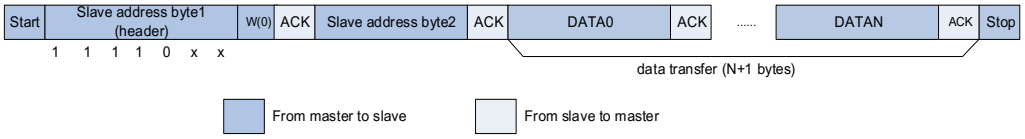
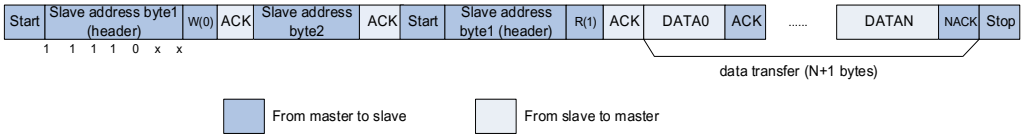


Figure 24-8. I2C communication flow with 10-bit address (Master Receive)



Programming model

An I2C device such as LCD driver may only be a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered as a slave.

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:

- Master Transmitter
- Master Receiver
- Slave Transmitter
- Slave Receiver

I2C block supports all of the four I2C modes. After system reset, it works in slave mode. After sending a START signal on I2C bus, it changes into master mode. The I2C changes back to slave mode after sending a STOP signal on I2C bus.

- **Programming model in slave transmitting mode**

As is shown in ***Figure 24-9. Programming model for slave transmitting (10-bit address mode)***, the following software procedure should be followed if users wish to transmit data in slave transmitter mode:

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.

2. After receiving a START signal followed by a matched address, either in 7-bit format or in 10-bit format, the I2C hardware sets the ADDSEND bit in I2C_STAT0 register, which should be monitored by software either by polling or interrupt. After that, software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. If 10-bit addressing format is selected, the I2C master should then send a repeated START signal followed by a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START signal and the following header. The ADDSEND bit must be cleared by software again by reading I2C_STAT0 and then I2C_STAT1.

3. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C_DATA are empty. Once TBE is set, software should write the first byte of data to I2C_DATA register, TBE is not cleared in this case because the byte written in I2C_DATA is moved to the internal shift register immediately. I2C begins to transmit data to I2C bus as soon as the shift register is not empty.

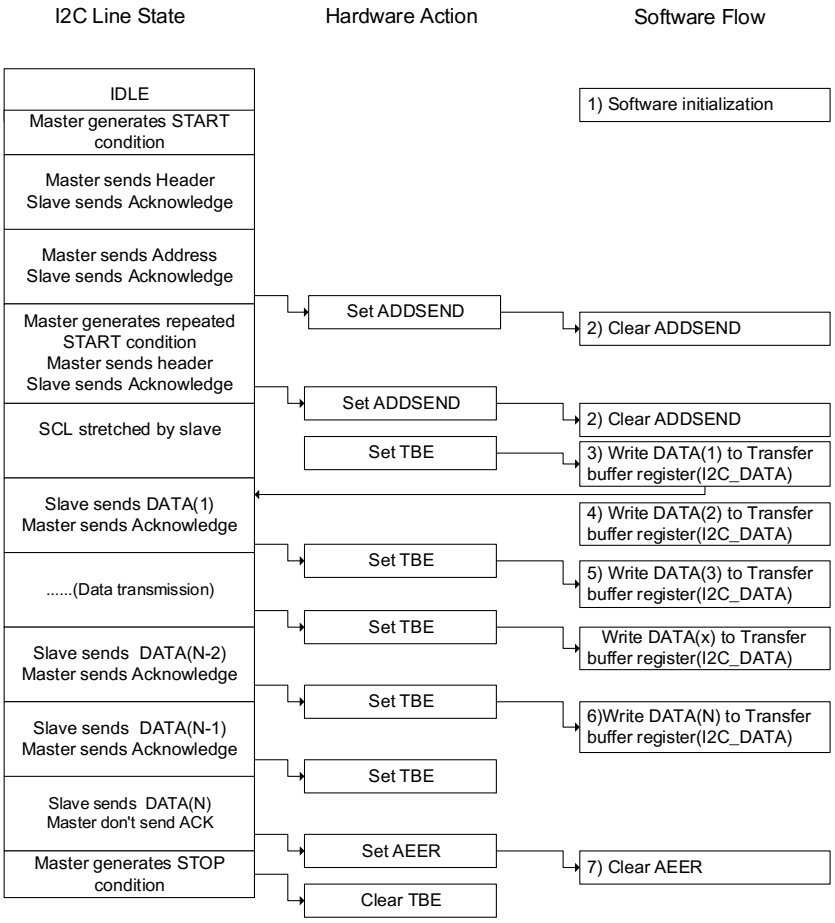
4. During the transmission of the first byte, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

5. After the transmission of the first byte, the TBE bit will be set, the software can write the third byte to the I2C_DATA register and TBE is cleared. After this, any time TBE is set, software can write a byte to I2C_DATA as long as there is still data to be transmitted.

6. During the transmission of the second last byte, software writes the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be set after the byte's transmission and not cleared until a STOP signal.

7. I2C master doesn't acknowledge to the last byte according to the I2C protocol, so after sending the last byte, I2C slave will wait for the STOP signal on I2C bus and sets AERR (Acknowledge Error) bit to notify software that the transmission completes. Software clears AERR bit by writing 0 to it.

Figure 24-9. Programming model for slave transmitting (10-bit address mode)



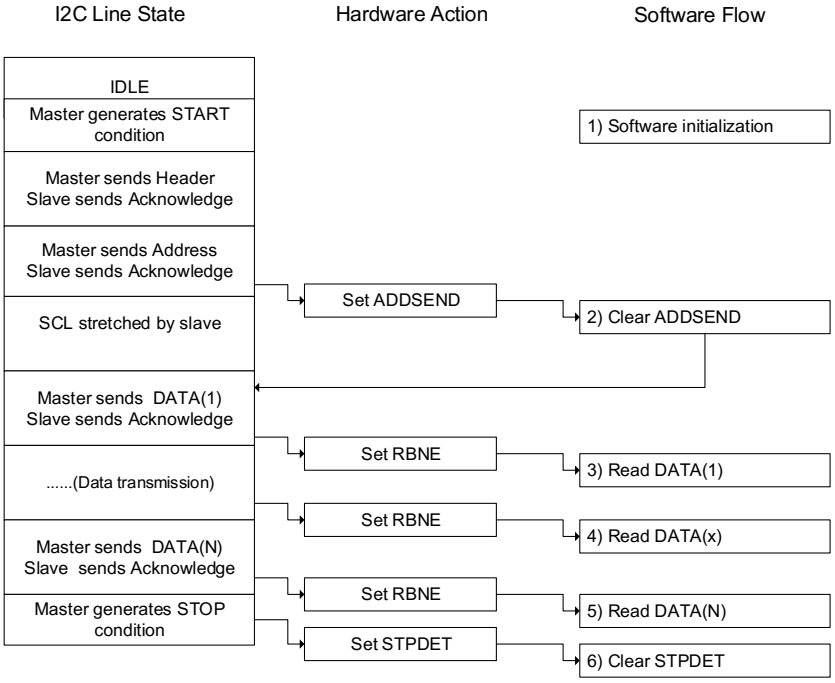
- **Programming model in slave receiving mode**

As is shown in ***Figure 24-10. Programming model for slave receiving (10-bit address mode)***, the following software procedure should be followed if users wish to receive data in slave receiver mode:

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. After receiving a START signal followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register 0, which should be monitored by software either by polling or interrupt. After that software should read I2C_STAT0 and then I2C_STAT1 to clear ADDSEND bit. The I2C begins to receive data on I2C bus as soon as ADDSEND bit is cleared.
3. As soon as the first byte is received, RBNE is set by hardware. Software can now read the first byte from I2C_DATA and RBNE is cleared as well.
4. Any time RBNE is set, software can read a byte from I2C_DATA.
5. After the last byte is received, RBNE is set. Software reads the last byte.

6. STPDET bit is set when I2C detects a STOP signal on I2C bus and software reads I2C_STAT0 and then writes I2C_CTL0 to clear the STPDET bit.

Figure 24-10. Programming model for slave receiving (10-bit address mode)



■ **Programming model in master transmitting mode**

As is shown in ***Figure 24-11. Programming model for master transmitting (10-bit address mode)***, the following software procedure should be followed if users wish to make transaction in master transmitter mode:

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address which has been sent is header of a 10-bit address, the hardware sets ADD10SEND bit after sending the header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.
4. After the 7-bit or 10-bit address has been sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1.
5. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C_DATA are empty. Software now writes the first byte

data to I2C_DATA register, but the TBE will not be cleared because the byte written in I2C_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as the shift register is not empty.

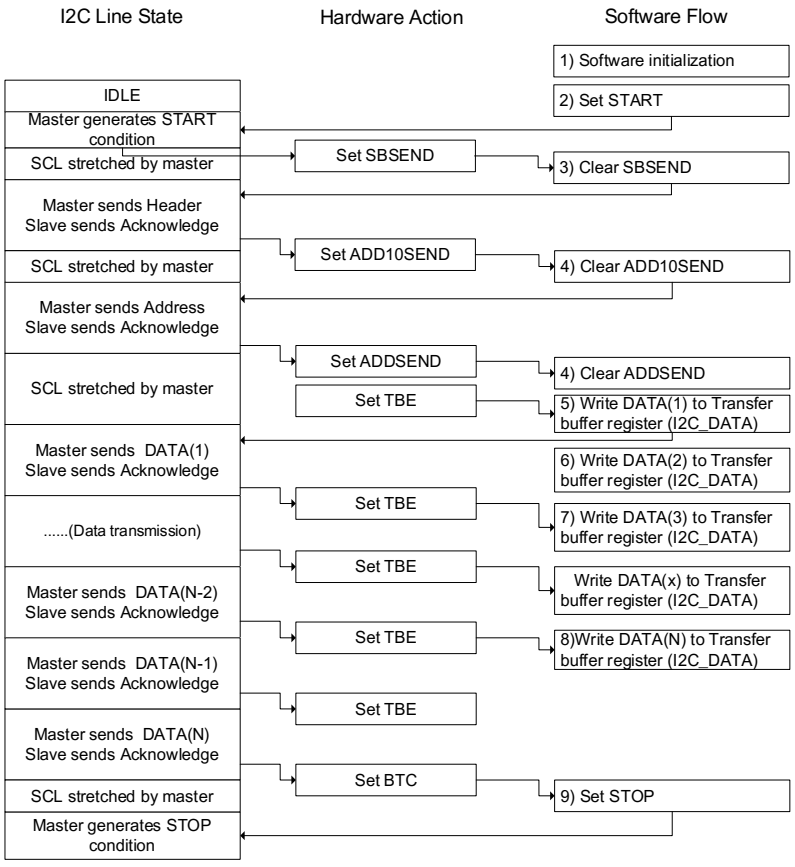
6. During the transmission of the first byte, software can write the second byte to I2C_DATA, and this time TBE is cleared because neither I2C_DATA nor shift register is empty.

7. Any time TBE is set, software can write a byte to I2C_DATA as long as there is still data to be transmitted.

8. During the transmission of the second last byte, software writes the last data to I2C_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be asserted after the transmission of the byte and not be cleared until a STOP signal.

9. After sending the last byte, I2C master sets BTC bit because both the shift register and I2C_DATA are empty. Software should set the STOP bit to generate a STOP signal, then the I2C clears both TBE and BTC flags.

Figure 24-11. Programming model for master transmitting (10-bit address mode)



■ **Programming model in master receiving mode**

In master receiving mode, a master is responsible for generating NACK for the last byte reception and then sending a STOP signal on I2C bus. So, special attention should be paid to ensure the correct ending of data reception. Two solutions for master receiving are provided here for applications: Solution A and B. Solution A requires the software's quick

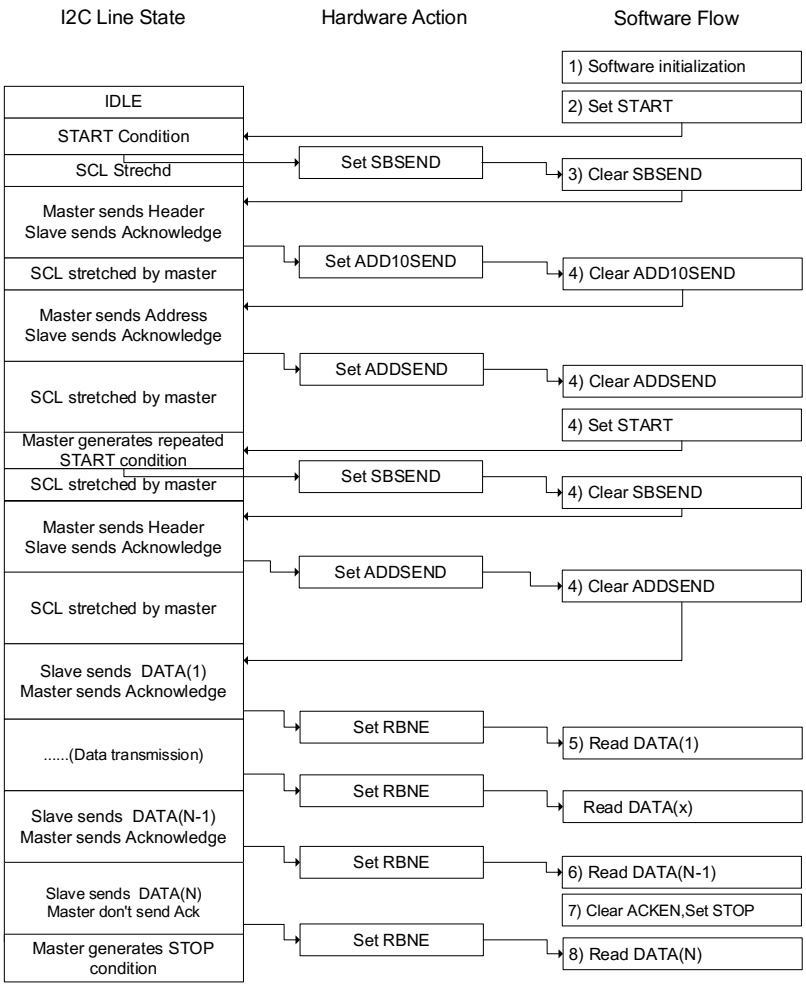
response to I2C events, while Solution B doesn't.

Solution A

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address which has been sent is header of a 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.
4. After the 7-bit or 10-bit address has been sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1. If the address is in 10-bit format, software should then set START bit again to generate a repeated START signal on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C_STAT0 and writing header to I2C_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C_STAT0 and then I2C_STAT1.
5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C_DATA.
7. After the second last byte (N-1) is received, the software should clear ACKEN bit and set STOP bit. These actions should complete before the end of the last byte's receiving to ensure that NACK will be sent for the last byte.
8. After the last byte is received, RBNE is set. Software reads the last byte. Since ACKEN has been cleared in the previous step, I2C doesn't send ACK for the last byte and it generates a STOP signal after the transmission of the last byte.

The above steps require byte number N>1. If N=1, Step 7 should be performed after Step 4 and completed before the end of the single byte's receiving.

Figure 24-12. Programming model for master receiving using Solution A (10-bit address mode)



Solution B

1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C_STAT0 register and enters master mode. Now software should clear the SBSEND bit by reading I2C_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address which has been sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C_STAT0 and writing 10-bit lower address to I2C_DATA.
4. After the 7-bit or 10-bit address has been sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C_STAT0 and then I2C_STAT1. If the address is in 10-bit format, software should then set START bit again to generate

a repeated START signal on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C_STAT0 and writing header to I2C_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C_STAT0 and then I2C_STAT1.

5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C_DATA and RBNE is cleared as well.

6. Any time RBNE is set, software can read a byte from I2C_DATA until the master receives N-3 bytes.

As shown in ***Figure 24-13. Programming model for master receiving mode using solution B (10-bit address mode)***, the N-2 byte is not read out by software, so after the N-1 byte is received, both BTC and RBNE are asserted. The bus is stretched by master to prevent the reception of the last byte. Then software should clear ACKEN bit.

7. Software reads out N-2 byte, clearing BTC. After this, the N-1 byte is moved from shift register to I2C_DATA and bus is released and begins to receive the last byte. Master doesn't send an ACK for the last byte because ACKEN is already cleared.

8. After the last byte is received, both BTC and RBNE are set again, and SCL is stretched low. Software sets STOP bit and master sends out a STOP signal on bus.

9. Software reads the N-1 byte, clearing BTC. After this the last byte is moved from shift register to I2C_DATA.

10. Software reads the last byte, clearing RBNE.

The above steps require that byte number N>2. N=1 and N=2 are similar:

N=1

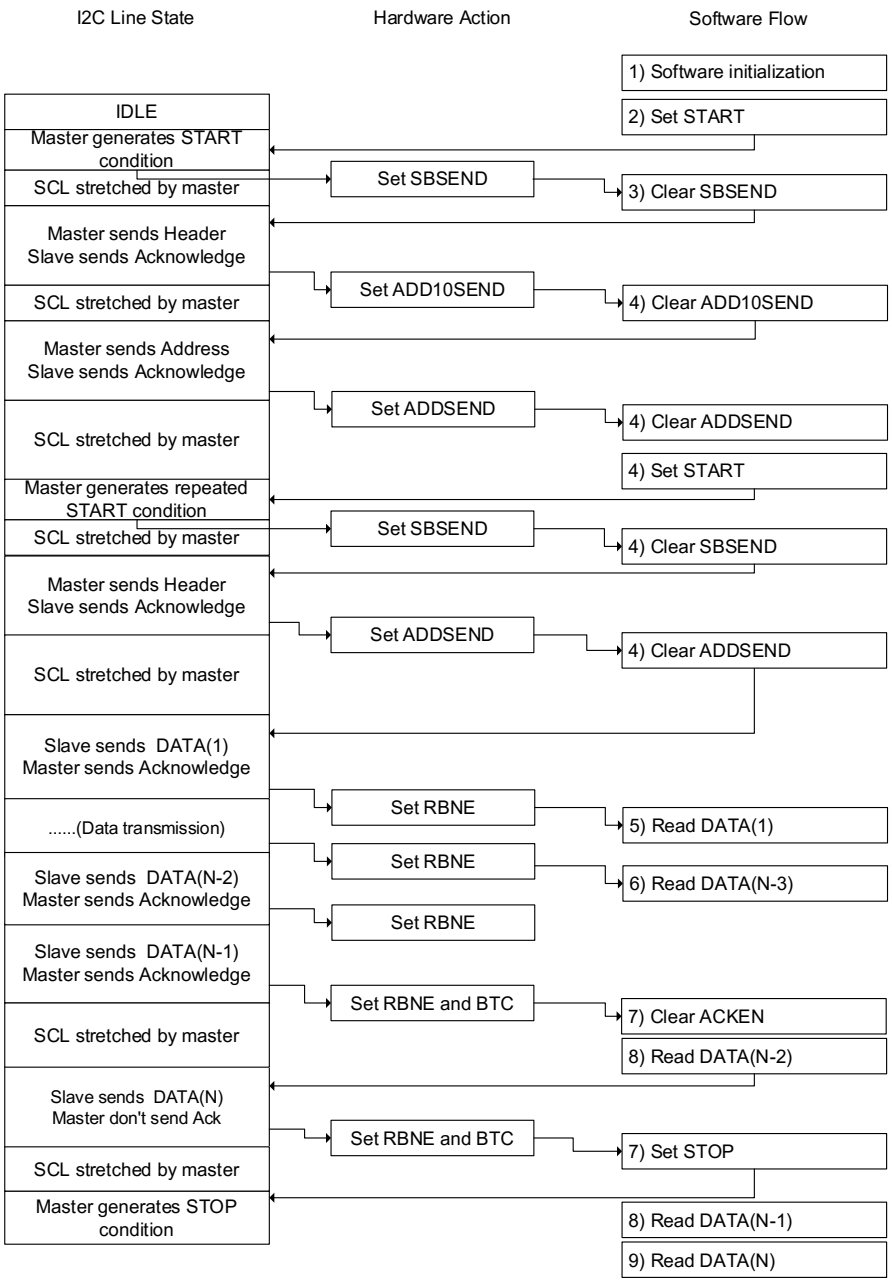
In Step4, software should reset ACKEN bit before clearing ADDSEND bit and set STOP bit after clearing ADDSEND bit. Step 5 is the last step when N=1.

N=2

In Step 2, software should set POAP bit before setting START bit. In Step 4, software should reset ACKEN bit before clearing ADDSEND bit. In Step 5, software should wait until BTC is set and then set STOP bit and read I2C_DATA twice.

Figure 24-13. Programming model for master receiving mode using solution B (10-bit

address mode)



SCL line stretching

The SCL line stretching function is designed to avoid overflow error in reception and underflow error in transmission. As is shown in Programming Model, when the TBE and BTC bits are set in transmitting mode, the transmitter stretches the SCL line low until the transfer buffer register is filled with the next data to be transmitted. When the RBNE and BTC bits are set in receiving mode, the receiver stretches the SCL line low until the data in the transfer buffer is read out.

When works in slave mode, the SCL line stretching function can be disabled by setting the

SS bit in the I2C_CTL0 register. If this bit is set, the software is required to be quick enough to serve the TBE, RBNE and BTC status, otherwise, overflow or underflow situation might occur.

Use DMA for data transfer

As is shown in Programming Model, each time TBE or RBNE is asserted, software should write or read a byte, this may cause CPU to be high overloaded. The DMA controller can be used to process TBE and RBNE flags: each time TBE or RBNE is asserted, DMA controller does a read or write operation automatically. It reduces the load on the CPU. See the DMA section for details on how to configure DMA.

The DMA request is enabled by the DMAON bit in the I2C_CTL1 register. This bit should be set after clearing the ADDSEND status. If the SCL line stretching function is disabled for a slave device, the DMAON bit should be set before the ADDSEND event.

Refer to the specification of the DMA controller for the configuration method of a DMA stream. The DMA controller must be configured and enabled before the I2C transfer. When the configured number of bytes have been transferred, the DMA controller generates End of Transfer (EOT) interrupt. DMA will send an End of Transmission (EOT) signal to the I2C interface and generates a DMA full transfer finish interrupt.

When a master receives two or more bytes, the DMALST bit in the I2C_CTL1 register should be set. The I2C master will send NACK after the last byte. The STOP bit can be set by software to generate a STOP signal in the ISR of the DMA full transfer finish interrupt.

When a master receives only one byte, the ACKEN bit must be cleared before clearing the ADDSEND status. Software can set the STOP bit to generate a STOP signal after clearing the ADDSEND status, or in the ISR of the DMA full transfer finish interrupt.

Packet error checking

There is a CRC-8 calculator in I2C block to perform PEC (Packet Error Checking) for I2C data. The polynomial of the CRC is x8 + x2 + x + 1 which is compatible with the SMBus protocol. If enabled by setting PECEN bit, the PEC will calculate all the data transmitted through I2C including address. I2C is able to send out the PEC value after the last data byte or check the received PEC value with its calculated PEC using the PECTRANS bit. In DMA mode, the I2C will send or check PEC value automatically if PECEN bit and PECTRANS bit are set.

Analog and digital noise filters

The I2C protocol requires to suppress spikes with length up to 50 ns on the SCL and SDA line in fast mode. There are two methods that can be used to fulfill this requirement: analog noise filter and digital noise filter.

The analog noise filter lies between the IO pins of SCL / SDA and the I2C digital logic. It is an analog block that can suppress spikes with length up to 50ns. The analog noise filter, which is enabled by default, can be disabled by setting the AFD bit in the I2C_FCTL register.

The digital noise filter is a digital block lies inside the I2C digital logic. It suppresses spikes with length up to (DF+1) PCLK cycles on the SCL / SDA inputs. If the analog filter is enabled, the input of the digital noise filter is the output of the analog noise filter.

The configuration of the analog and digital noise filters can only be changed when I2C is disabled.

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 transmission 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 is realized based on I2C hardware and it uses I2C hardware addressing, but it adds the 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 allows 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.

- **Time-out feature**

  SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency is 10 kHz to prevent locking up the bus. I2C can be a 'DC' bus, which means that a slave device stretches the master clock when performing some routines while the master is accessing it. This will notify the master that the slave is busy but does not want to lose the communication. The slave device will continue the communication after its task is completed. There is no limit in the I2C bus protocol of how long this delay can be, whereas for a SMBus system, it would be limited to 25ms. 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 solve the problem. Slave devices are not allowed to hold the clock low too long.

- **Packet error checking**

SMBus 2.0 and 1.1 allow Packet Error Checking (PEC). In that mode, a PEC byte is appended at the end of each transaction. The byte is a CRC-8 checksum of the entire message including the address and read/write bit. The polynomial used is x8+x2+x+1 (the CRC-8-ATM HEC algorithm, initialized to zero).

- **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. SMBus also defines a less common "Host Notify Protocol", providing similar notifications which is based on the I2C multi-master mode but it can pass more data.

Note: When using SMBus host mode, the I2C_SMBA pin is checked by default and assigned as AFIO (I2C0 corresponds to PB5, I2C1 corresponds to PB12, I2C2 corresponds to PA9). An interrupt will be generated when error interrupts are enabled and the I2C_SMBA pin is at a low level. Therefore, when SMBus alerts are not needed, the use of the I2C_SMBA pin should be avoided, and the corresponding IO should be kept at a high level.

- **SMBus programming flow**

The programming flow for SMBus is similar to normal I2C. In order to use SMBus mode, the application should configure several SMBus specific registers, respond to some SMBus specific flags and implement the upper protocols described in SMBus specification.

1. Before communication, SMBEN bit in I2C_CTL0 should be set and SMBSEL and ARPEN bits should be configured to desired values.
2. In order to support address resolution protocol (ARP) (ARPEN=1), the software should respond to HSTSMB flag in SMBus Host Mode (SMBSEL =1) or DEFSMB flag in SMBus Device Mode, and implement the function of ARP protocol.
3. In order to support SMBus Alert Mode, the software should respond to SMBALT flag and implement the related function.

SAM_V support

To support the SAM_V standard, two additional pins are added to the I2C module: txframe and rxframe. Txframe is an output pin, in master mode, it indicates the I2C is busy when it is asserted. Rxframe is an input pin that is supposed to be multiplexed together with the SMBALERT signal.

The SAM_V mode is enabled by setting the SAMEN bit of the I2C_SAMCS register. The status of the txframe and rxframe pin can be reflected by the RFR, RFF, TFR, TFF, RXF, and TXF flags of the I2C_SAMCS register. I2C interrupts will be generated if the corresponding interrupt enable bits are set.

Status, errors and interrupts

There are several status and error flags in I2C, and interrupts may be asserted from these flags by setting some register bits (refer to ***Register definition*** for detail).

Table 24-2. Event status flags

Event Flag Name	Description
SBSEND	START signal sent (master)
ADDSEND	Address sent or received
ADD10SEND	Header of 10-bit address sent
STPDET	STOP signal detected
BTC	Byte transmission completed
TBE	I2C_DATA is empty when transmitting
RBNE	I2C_DATA is not empty when receiving
RFR	SAM_V mode rxframe pin rising edge is detected
RFF	SAM_V mode rxframe pin falling edge is detected
TFR	SAM_V mode txframe pin rising edge is detected
TFF	SAM_V mode txframe pin falling edge is detected

Table 24-3. Error flags

Error Name	Description
BERR	Bus error
LOSTARB	Arbitration lost
OUERR	Over-run or under-run when SCL stretch is disabled.
AERR	No acknowledge received
PECERR	CRC value doesn't match
SMBTO	Bus timeout in SMBus mode
SMBALT	SMBus Alert

24.1.4. Register definition

I2C0 base address: 0x4000 5400

I2C1 base address: 0x4000 5800

I2C2 base address: 0x4000 5C00

Control register 0 (I2C_CTL0)

Address offset: 0x00
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
SRESET	Reserved	SALT	PECTRANS	POAP	ACKEN	STOP	START	SS	GCEN	PECEN	ARPEN	SMBSEL	Reserved	SMBEN	I2CEN
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	SRESET	Software resets I2C, software should wait until the I2C lines are released to reset the I2C. 0: I2C is not under reset 1: I2C is under reset
14	Reserved	Must be kept at reset value.
13	SALT	SMBus Alert. Issue alert through SMBA pin. Software can set and clear this bit and hardware can clear this bit. 0: Don't issue alert through SMBA pin 1: Issue alert through SMBA pin
12	PECTRANS	PEC transfer Software sets and clears this bit while hardware clears this bit when PEC is transferred or START / STOP signal is detected or I2CEN=0. 0: Don't transfer PEC value 1: Transfer PEC value
11	POAP	Position of ACK and PEC when receiving This bit is set and cleared by software and cleared by hardware when I2CEN=0 0: ACKEN bit specifies whether to send ACK or NACK for the current byte that is being received. PECTRANS bit indicates that the current receiving byte is a PEC

		byte 1: ACKEN bit specifies whether to send ACK or NACK for the next byte that is to be received, PECTRANS bit indicates the next byte that is to be received is a PEC byte
10	ACKEN	Whether or not to send an ACK This bit is set and cleared by software and cleared by hardware when I2CEN=0 0: ACK will not be sent 1: ACK will be sent
9	STOP	Generate a STOP signal on I2C bus This bit is set and cleared by software and set by hardware when SMBus timeout and cleared by hardware when STOP signal is detected. 0: STOP will not be sent 1: STOP will be sent
8	START	Generate a START signal on I2C bus This bit is set and cleared by software and cleared by hardware when a START signal is detected or I2CEN=0. 0: START will not be sent 1: START will be sent
7	SS	Whether to stretch SCL low when data is not ready in slave mode. This bit is set and cleared by software. 0: SCL stretching is enabled 1: SCL stretching is disabled
6	GCEN	Whether or not to response to a General Call (0x00) 0: Slave won't respond to a General Call 1: Slave will respond to a General Call
5	PECEN	PEC calculation enable 0: PEC calculation disable 1: PEC calculation enable
4	ARPEN	ARP protocol enable in SMBus mode 0: ARP is disabled 1: ARP is enabled
3	SMBSEL	SMBus type selection 0: Device 1: Host
2	Reserved	Must be kept at reset value.
1	SMBEN	SMBus/I2C mode switch 0: I2C mode 1: SMBus mode
0	I2CEN	I2C peripheral enable 0: I2C is disabled

1: I2C is enabled

Control register 1 (I2C_CTL1)

Address offset: 0x04

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
RBNECM	Reserved		DMALST	DMAON	BUFIE	EVIE	ERRIE	Reserved		I2CCLK[5:0]					
rw			rw	rw	rw	rw	rw	rw							

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	RBNECM	RBNE clear condition control 0: RBNE can be cleared when I2C_DATA is read and BTC is cleared 1: RBNE can be cleared when I2C_DATA is read
14:13	Reserved	Must be kept at reset value.
12	DMALST	DMA last transfer configure 0: Next DMA EOT is not the last transfer 1: Next DMA EOT is the last transfer
11	DMAON	DMA mode switch 0: DMA mode switched off 1: DMA mode switched on
10	BUFIE	Buffer interrupt enable 0: Buffer interrupt is disabled, TBE = 1 or RBNE = 1 when EVIE=1 will not generate an interrupt. 1: Buffer interrupt is enabled, which means that interrupt will be generated when TBE = 1 or RBNE = 1 if EVIE=1.
9	EVIE	Event interrupt enable 0: Event interrupt is disabled 1: Event interrupt is enabled, which means that interrupt will be generated when SBSEND, ADDSEND, ADD10SEND, STPDET or BTC flag asserted or TBE=1 or RBNE=1 if BUFIE=1.
8	ERRIE	Error interrupt enable 0: Error interrupt is disabled 1: Error interrupt is enabled, which means that interrupt will be generated when BERR, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALT flag is asserted.

7:6	Reserved	Must be kept at reset value.
5:0	I2CCLK[5:0]	I2C peripheral clock frequency I2CCLK [5:0] should be the frequency of input APB1 clock in MHz which is at least 2. 000000 - 000001: Not allowed 000010 - 110010: 2 MHz~50MHz 110011 - 111111: Not allowed due to the limitation of APB1 clock **Note:** In I2C standard mode, the frequencies of APB1 must be equal or greater than 2MHz. In I2C fast mode, the frequencies of APB1 must be equal or greater than 8MHz.

Slave address register 0 (I2C_SADDR0)

Address offset: 0x08
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
ADDFORMAT	Reserved					ADDRESS[9:8]		ADDRESS[7:1]							ADDRESS0
rw						rw		rw							rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	ADDFORMAT	Address format for the I2C slave 0: 7-bit address 1: 10-bit address
14:10	Reserved	Must be kept at reset value.
9:8	ADDRESS[9:8]	Highest two bits of a 10-bit address
7:1	ADDRESS[7:1]	7-bit address or bits 7:1 of a 10-bit address
0	ADDRESS0	Bit 0 of a 10-bit address

Slave address register 1 (I2C_SADDR1)

Address offset: 0x0C
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								ADDRESS2[7:1]							DUADEN
								rw							rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:1	ADDRESS2[7:1]	The second I2C address for the slave in Dual-Address mode
0	DUADEN	Dual-Address mode enable 0: Dual-Address mode is disabled 1: Dual-Address mode is enabled

Transfer buffer register (I2C_DATA)

Address offset: 0x10

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								TRB[7:0]							
								rw							

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	TRB[7:0]	Transmission or reception data buffer

Transfer status register 0 (I2C_STAT0)

Address offset: 0x14

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
SMBALT	SMBTO	Reserved	PECERR	OUERR	AERR	LOSTARB	BERR	TBE	RBNE	Reserved	STPDET	ADD10SEND	BTC	ADDSEND	SBSEND
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	SMBALT	SMBus Alert status This bit is set by hardware and cleared by writing 0. 0: SMBA pin not pulled down (device mode) or no Alert detected (host mode) 1: SMBA pin pulled down and Alert address received (device mode) or Alert detected (host mode)
14	SMBTO	Timeout signal in SMBus mode This bit is set by hardware and cleared by writing 0. 0: No timeout error 1: Timeout event occurs (SCL is low for 25 ms)
13	Reserved	Must be kept at reset value.
12	PECERR	PEC error when receiving data This bit is set by hardware and cleared by writing 0. 0: Received PEC matches the calculated PEC 1: Received PEC doesn't match the calculated PEC, I2C will send NACK careless of ACKEN bit.
11	OUERR	Over-run or under-run situation occurs in slave mode, when SCL stretching is disabled. In slave receiving mode, if the last byte in I2C_DATA is not read out while the following byte is already received, over-run occurs. In slave transmitting mode, if the current byte is already sent out, while the I2C_DATA is still empty, under-run occurs. This bit is set by hardware and cleared by writing 0. 0: No over-run or under-run occurs 1: Over-run or under-run occurs
10	AERR	Acknowledge error This bit is set by hardware and cleared by writing 0. 0: No acknowledge error 1: Acknowledge error
9	LOSTARB	Arbitration lost in master mode This bit is set by hardware and cleared by writing 0. 0: No arbitration lost 1: Arbitration lost occurs and the I2C block changes back to slave mode.
8	BERR	Bus error A bus error occurs when an unexpected START or STOP signal on I2C bus This bit is set by hardware and cleared by writing 0. 0: No bus error 1: A bus error detected
7	TBE	I2C_DATA is empty during transmitting This bit is set by hardware after it moves a byte from I2C_DATA to shift register and

		cleared by writing a byte to I2C_DATA. If both the shift register and I2C_DATA are empty, writing I2C_DATA won't clear TBE (refer to Programming Model for detail). 0: I2C_DATA is not empty 1: I2C_DATA is empty, software can write
6	RBNE	I2C_DATA is not empty during receiving This bit is set by hardware after it moves a byte from shift register to I2C_DATA and cleared by reading I2C_DATA. If both BTC and RBNE are asserted, reading I2C_DATA won't clear RBNE because the byte in shift register will be moved to I2C_DATA immediately. 0: I2C_DATA is empty 1: I2C_DATA is not empty, software can read
5	Reserved	Must be kept at reset value.
4	STPDET	STOP signal is detected in slave mode This bit is set by hardware and cleared by reading I2C_STAT0 and then writing I2C_CTL0. 0: STOP signal not detected in slave mode 1: STOP signal detected in slave mode
3	ADD10SEND	Header of 10-bit address is sent in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA. 0: No header of 10-bit address is sent in master mode 1: Header of 10-bit address is sent in master mode
2	BTC	Byte transmission is completed. If a byte is already received in shift register but I2C_DATA is still full in receiving mode or a byte is already sent out from shift register but I2C_DATA is still empty in transmitting mode, the BTC flag is asserted if SCL stretching enabled. This bit is set by hardware and cleared by 3 ways as follow: 1. Software clearing: reading I2C_STAT0 followed by reading or writing I2C_DATA 2. Hardware clearing: sending the STOP signal or START signal 3. Bit 0 (I2CEN bit) of the I2C_CTL0 is reset. 0: BTC not asserted 1: BTC asserted
1	ADDSEND	Address is sent and ACK is received in master mode or address is received and matches its own address in slave mode. This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1. 0: In slave mode, no address is received or the received address does not match with its own address. In master mode, no address is sent or address has been sent but not received the ACK from slave. 1: In slave mode, address is received and matches with its own address. In master mode, address has been sent and receives the ACK from slave.

0	SBSEND	START signal is sent out in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA. 0: No START signal sent 1: START signal sent

Transfer status register 1 (I2C_STAT1)

Address offset: 0x18

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
PECV[7:0]								DUMODF	HSTSMB	DEFSMB	RXGC	Reserved	TR	I2CBSY	MASTER
r								r	r	r	r		r	r	r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:8	PECV[7:0]	Packet Error Checking value that calculated by hardware when PEC is enabled.
7	DUMODF	Dual flag in slave mode indicates which address matches with the address in Dual-Address mode This bit is cleared by hardware after a STOP or a START signal or I2CEN=0 0: The address matches with SADDR0 address 1: The address matches with SADDR1 address
6	HSTSMB	SMBus host header detected in slave mode This bit is cleared by hardware after a STOP or a START signal or I2CEN=0 0: No SMBus host header is detected 1: SMBus host header is detected
5	DEFSMB	Default address of SMBus device This bit is cleared by hardware after a STOP or a START signal or I2CEN=0. 0: The default address has not been received for SMBus device 1: The default address has been received for SMBus device
4	RXGC	General call address (0x00) received. This bit is cleared by hardware after a STOP or a START signal or I2CEN=0. 0: No general call address received 1: General call address received
3	Reserved	Must be kept at reset value.
2	TR	Transmitter or receiver

		This bit indicates whether the I2C is a transmitter or a receiver. It is cleared by hardware after a STOP or a START signal or I2CEN=0 or LOSTARB=1. 0: Receiver 1: Transmitter
1	I2CBSY	Busy flag This bit is cleared by hardware after a STOP signal 0: No I2C communication. 1: I2C communication active.
0	MASTER	A flag indicating whether I2C block is in master or slave mode. This bit is set by hardware when a START signal generates. This bit is cleared by hardware after a STOP signal or I2CEN=0 or LOSTARB=1. 0: Slave mode 1: Master mode

Clock configure register (I2C_CKCFG)

Address offset: 0x1C

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
FAST	DTCY	Reserved		CLKC[11:0]											
rw	rw			rw											

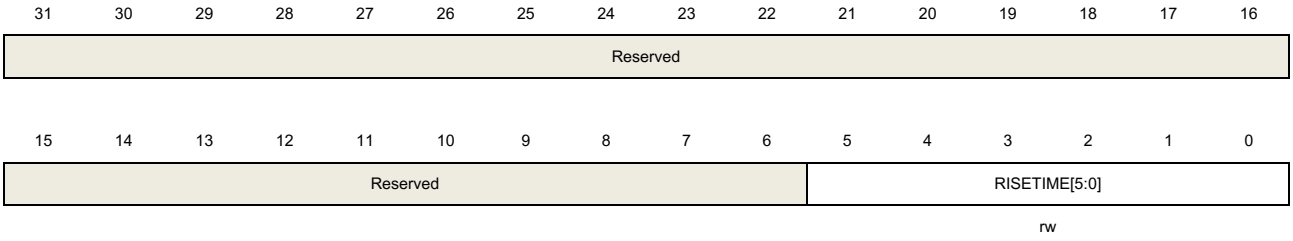
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	FAST	I2C speed selection in master mode 0: Standard speed 1: Fast speed
14	DTCY	Duty cycle in fast mode 0: $T_{low}/T_{high}=2$ 1: $T_{low}/T_{high}=16/9$
13:12	Reserved	Must be kept at reset value.
11:0	CLKC[11:0]	I2C clock control in master mode In standard speed mode: $T_{high}=T_{low}=CLKC*T_{PCLK1}$ In fast speed mode, if DTCY=0: $T_{high}=CLKC*T_{PCLK1}$, $T_{low}=2*CLKC*T_{PCLK1}$ In fast speed mode, if DTCY=1: $T_{high}=9*CLKC*T_{PCLK1}$, $T_{low}=16*CLKC*T_{PCLK1}$

Note: If DTCY is 0, when PCLK1 is an integral multiple of 3, the baud rate will be more accurate. If DTCY is 1, when PCLK1 is an integral multiple of 25, the baud rate will be more accurate.

Rise time register (I2C_RT)

Address offset: 0x20
Reset value: 0x0000 0002

This register can be accessed by half-word (16-bit) or word (32-bit).

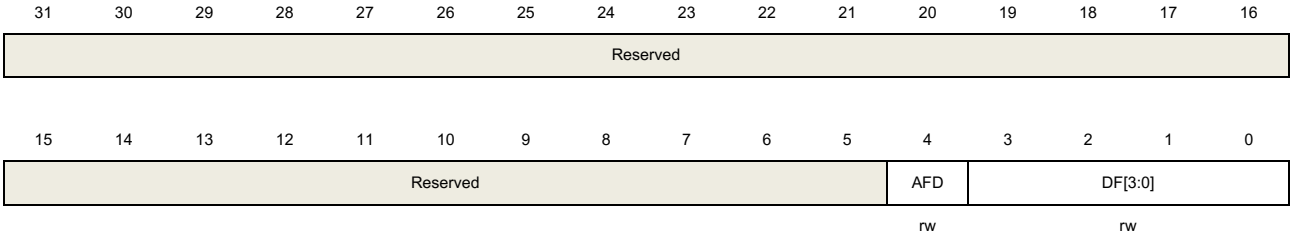


Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5:0	RISETIME[5:0]	Maximum rise time in master mode The RISETIME value should be the maximum SCL rise time incremented by 1.

Filter control register (I2C_FCTL)

Address offset: 0x24
Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit).



Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	AFD	Analog noise filter disable 0: Enable analog noise filter 1: Disable analog noise filter
3:0	DF[3:0]	Digital noise filter The Digital noise filter can filter spikes with the maximum length of DF [3:0] PCLK1 cycles on the input pins: SCL and SDA. 0000: Disable Digital noise filter

0001: Enable Digital noise filter and the maximum length of filtered spike up to 1 PCLK1 cycles.

...

1111: Enable Digital noise filter and the maximum length of filtered spike up to 15 PCLK1 cycles.

SAM control and status register (I2C_SAMCS)

Address offset: 0x80

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
RFR	RFF	TFR	TFF	Reserved		RXF	TXF	RFRIE	RFFIE	TFRIE	TFFIE	Reserved		STOEN	SAMEN
rc_w0	rc_w0	rc_w0	rc_w0			r	r	rw	rw	rw	rw			rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	RFR	Rxframe rise flag, cleared by software by writing 0
14	RFF	Rxframe fall flag, cleared by software by writing 0
13	TFR	Txframe rise flag, cleared by software by writing 0
12	TFF	Txframe fall flag, cleared by software by writing 0
11:10	Reserved	Must be kept at reset value.
9	RXF	Level of rxframe signal
8	TXF	Level of txframe signal
7	RFRIE	Rxframe rise interrupt enable 0: Rxframe rise interrupt disabled 1: Rxframe rise interrupt enabled
6	RFFIE	Rxframe fall interrupt enable 0: Rxframe fall interrupt disabled 1: Rxframe fall interrupt enabled
5	TFRIE	Txframe rise interrupt enable 0: Txframe rise interrupt disabled 1: Txframe rise interrupt enabled
4	TFFIE	Txframe fall interrupt enable 0: Txframe fall interrupt disabled

		1: Txframe fall interrupt enabled
3:2	Reserved	Must be kept at reset value.
1	STOEN	SAM_V interface timeout detect enable 0: SAM_V interface timeout detect disabled 1: SAM_V interface timeout detect enabled
0	SAMEN	SAM_V interface enable 0: SAM_V interface disabled 1: SAM_V interface enabled

24.2. Inter-integrated circuit interface (I2Cx, x=3, 4, 5)

24.2.1. 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.

24.2.2. 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 address, 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, I2CxFMP (x = 3,4,5) must be enabled in SYSCFG_CFG1).
- 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.

24.2.3. Function overview

Figure 24-14. I2C module block diagram below provides details on the internal configuration of the I2C interface.

Figure 24-14. I2C module block diagram

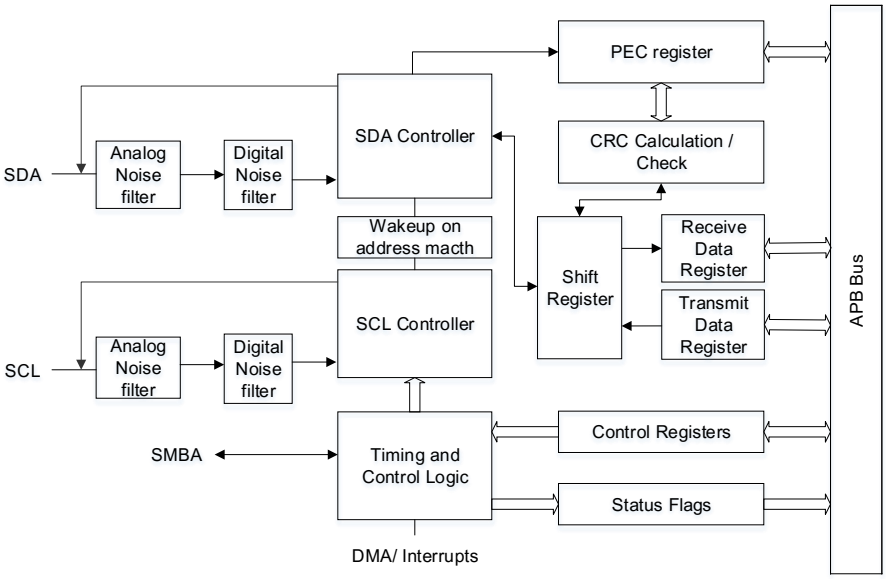


Table 24-4. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)

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

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)
- PLLSAIR: Phase Lock Loop
- IRC16M: Internal 16M 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 260ns. 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}.

I2C communication flow

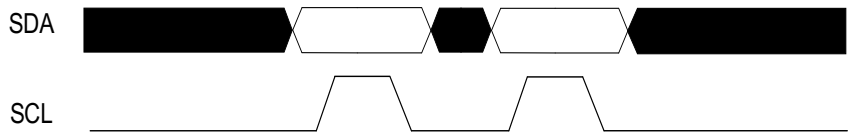
An I2C device is able to transmit or receive data whether it's a master or a slave, thus, there're 4 operation modes for an I2C device:

- 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 24-15. Data validation*). One clock pulse is generated for each data bit transferred.

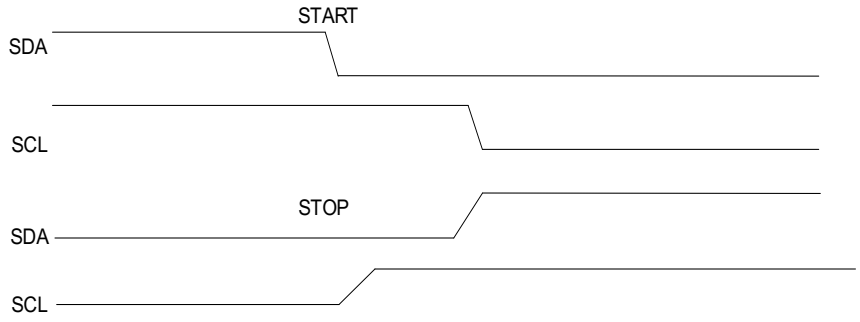
Figure 24-15. Data validation



- **START and STOP signal**

All transactions begin with a START and are terminated by a STOP (see *Figure 24-16. START and STOP signal*). 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 24-16. START and STOP signal



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 signal on 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 signal contain 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 signal and it's also responsible for SCL clock generation.

In master mode, if AUTOEND=1, the STOP signals generated automatically by hardware. If AUTOEND=0, the STOP signal generated by software, or the master can generate a RESTART signal to start a new transfer.

Figure 24-17. I2C communication flow with 10-bit address (Master Transmit)

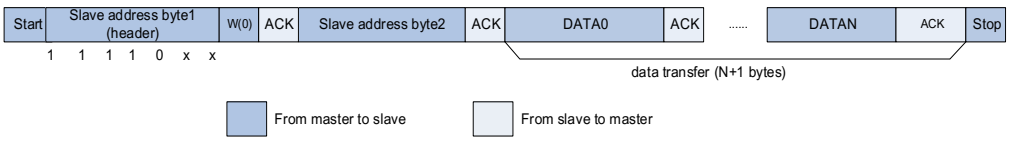


Figure 24-18. I2C communication flow with 7-bit address (Master Transmit)

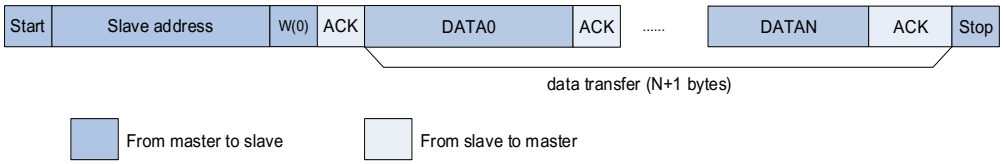
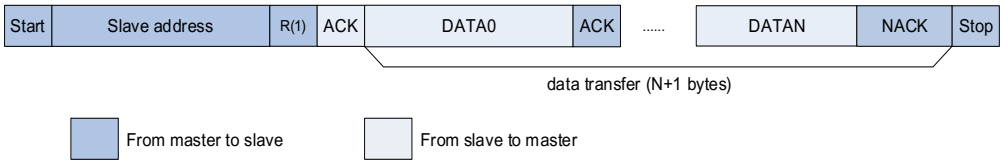


Figure 24-19. I2C communication flow with 7-bit address (Master Receive)



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 executed 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 24-20. 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 24-21. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)***.

Figure 24-20. I2C communication flow with 10-bit address (Master Receive when HEAD10R=0)

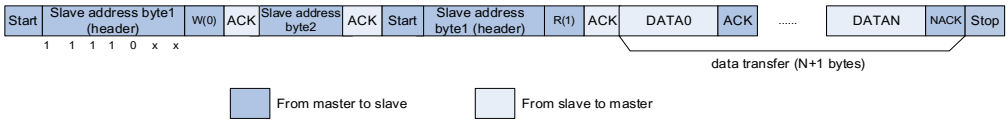
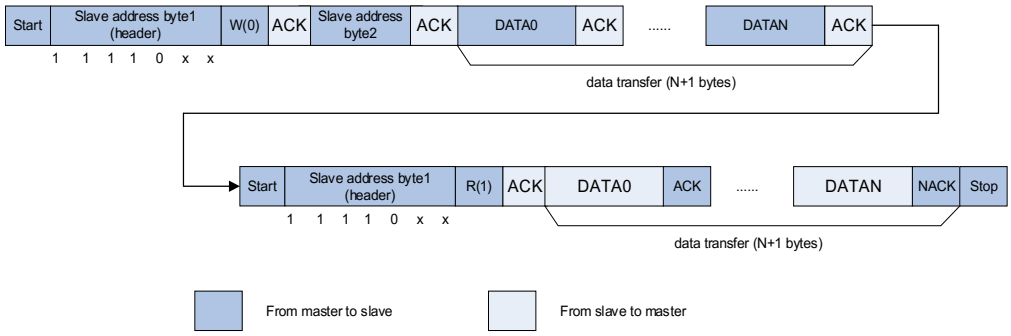


Figure 24-21. I2C communication flow with 10-bit address (Master Receive when HEAD10R=1)



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 50ns in fast mode and fast mode plus.

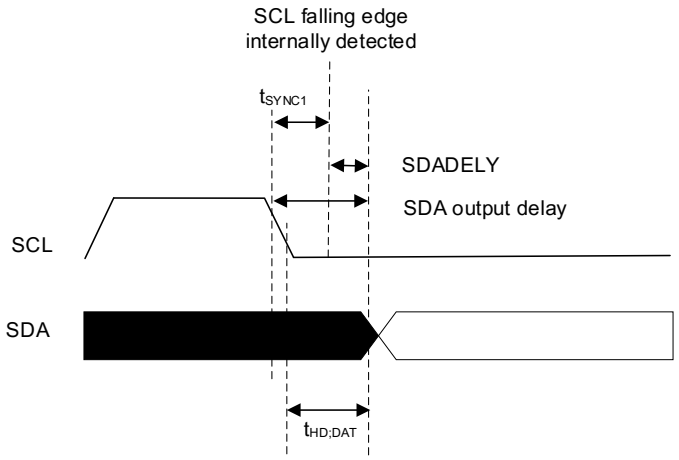
The digital noise filter can be used by configuring the DNF [3:0] bit in I2C_CTL0 register. The level of the SCL or the SDA will not be changed if the level is stable for no more than DNF[3:0]×t_{I2CCLK}. The length of spikes to be suppressed is configured by DNF[3:0].

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.

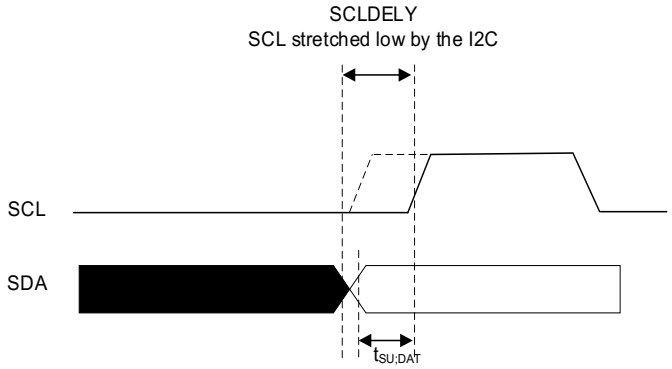
If the data is already available in I2C_TDATA register, the data will be sent on SDA after the SDADELY delay. As is shown in ***Figure 24-22. Data hold time***.

Figure 24-22. Data hold time



The SCLDELY counter starts when the data is sent on SDA output. As is shown in ***Figure 24-23. Data setup time***.

Figure 24-23. 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 + 1)* (PSC+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 24-5. 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	

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:

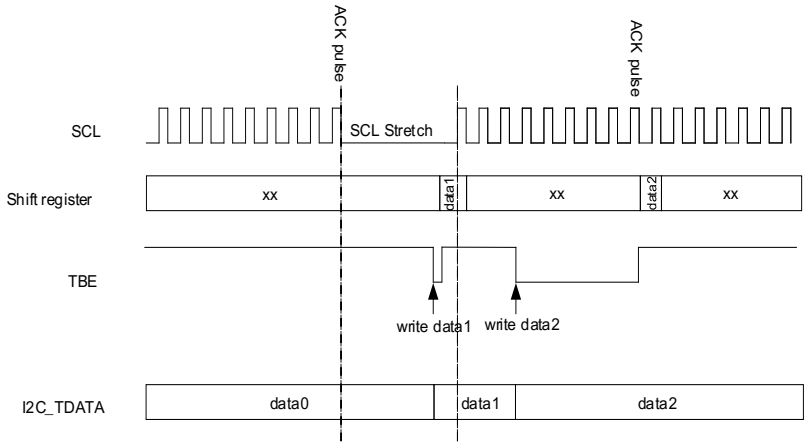
- Write I2CEN = 0
- Check I2CEN = 0
- Write I2CEN = 1

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.

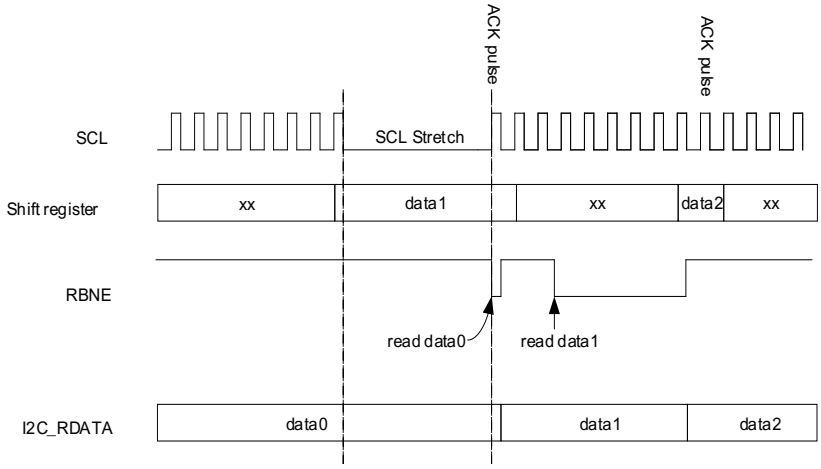
Figure 24-24. Data transmission



- **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 will be stretched until the previous received data in I2C_RDATA is read. The stretch is inserted before the acknowledge pulse.

Figure 24-25. 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 24-6. Communication modes to be shut down***, the I2C embedded a byte counter in the hardware.

Table 24-6. Communication modes to be shut down

Working mode	Action
Master mode	NACK, STOP and RESTART generation
Slave receiver mode	ACK control
SMBus mode	PEC generation/checking

The number of bytes to be transferred is configured by BYTENUM [7:0] in I2C_CTL1 register. If BYTENUM is 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.

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 must 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.

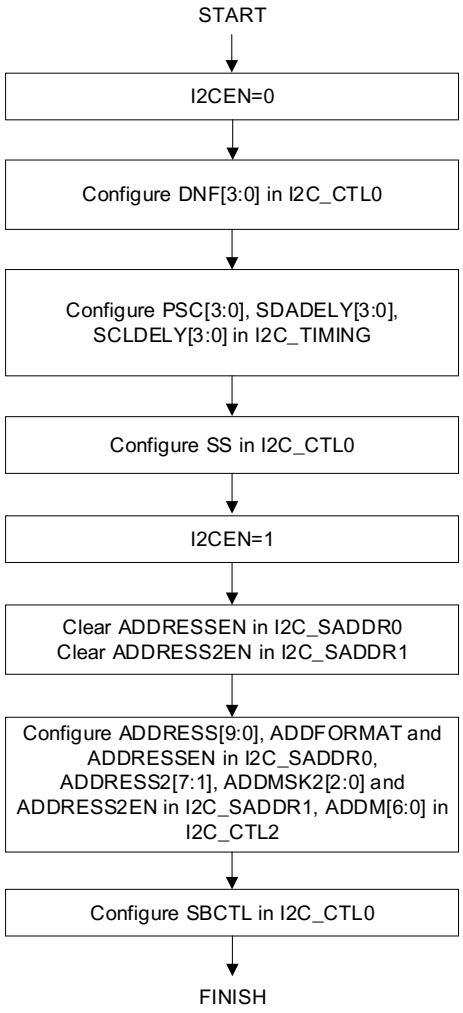
When the BYTENUM [7:0] is greater than 0x1, there is no stretch between the reception of two data bytes.

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 24-26. 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 chosen 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 and an underrun error occurs.
- 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 24-27. Programming model for slave transmitting when SS=0

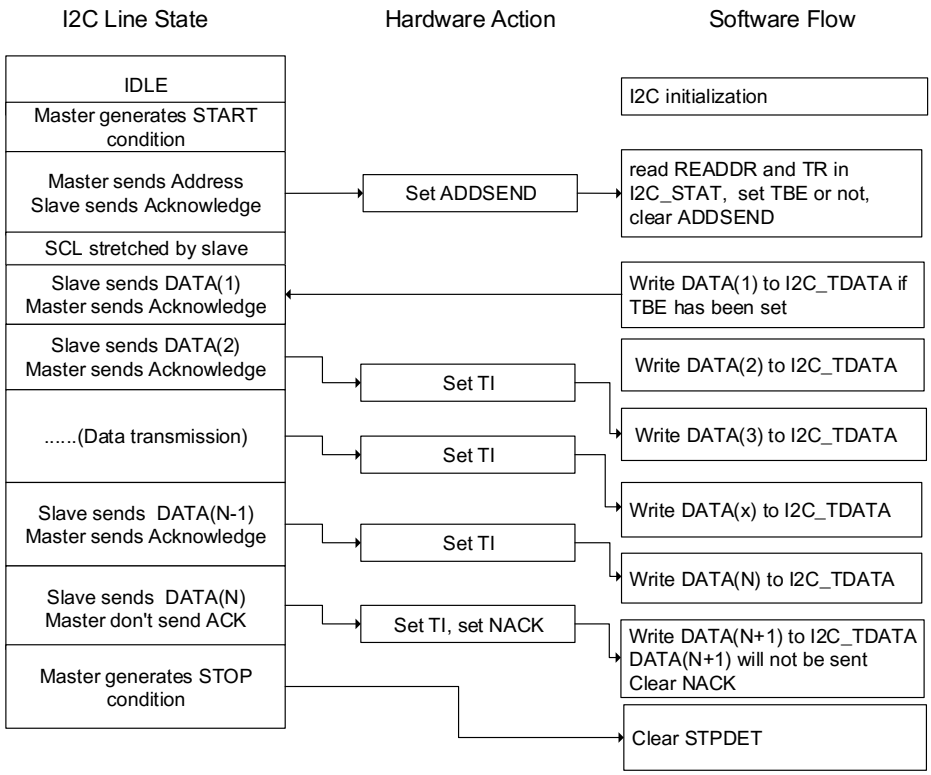
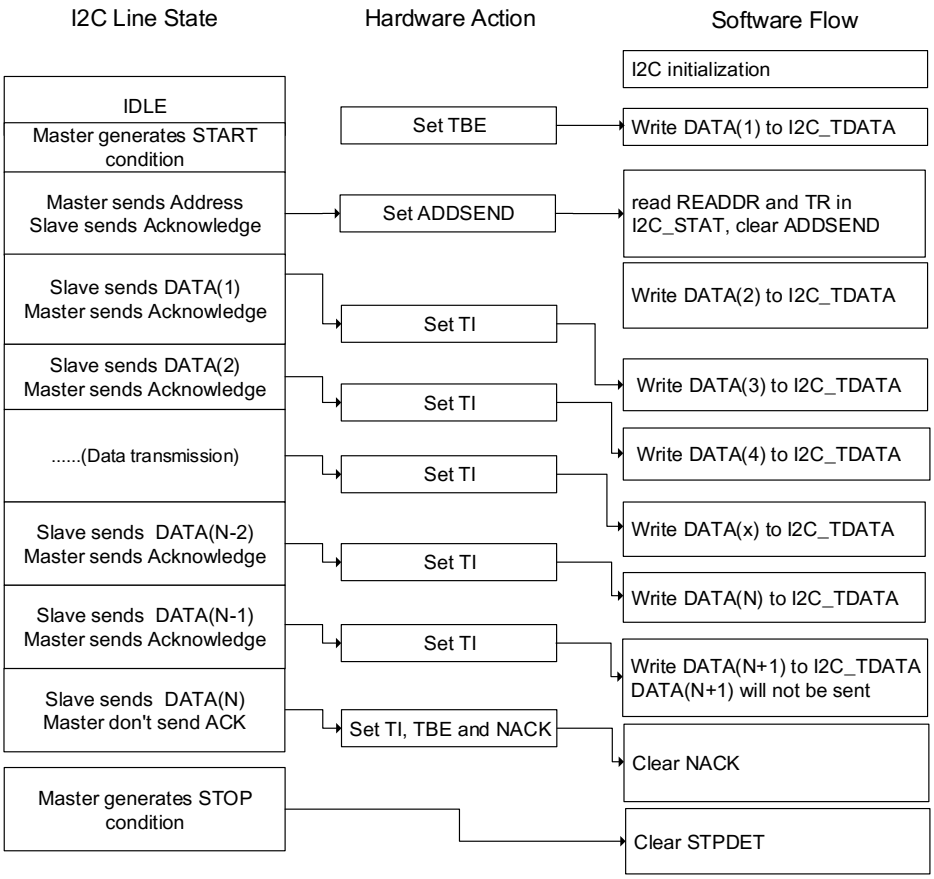


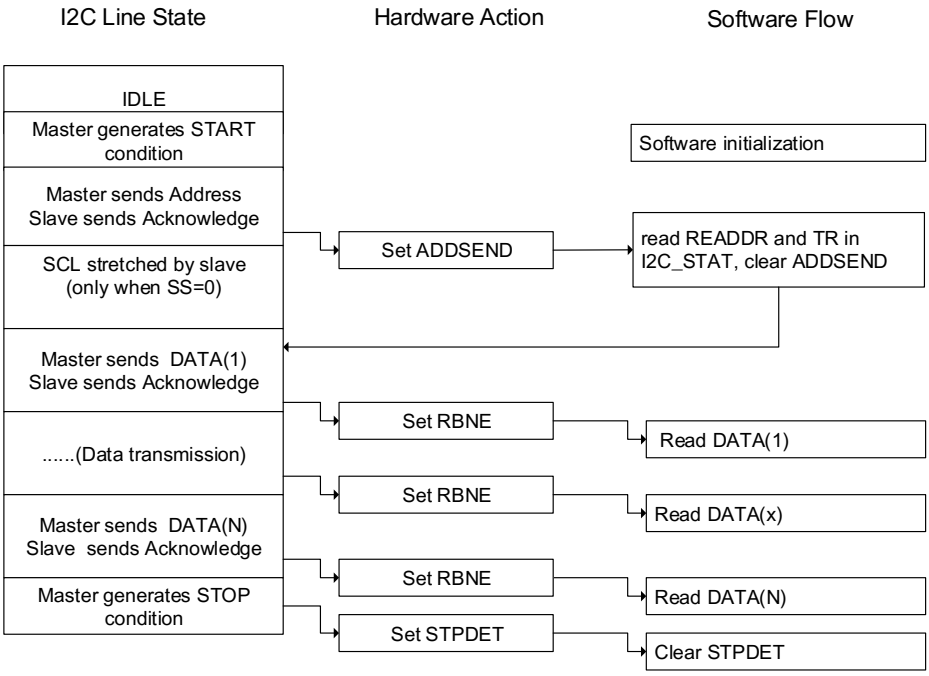
Figure 24-28. Programming model for slave transmitting when SS=1



- **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 24-29. Programming model for slave receiving



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

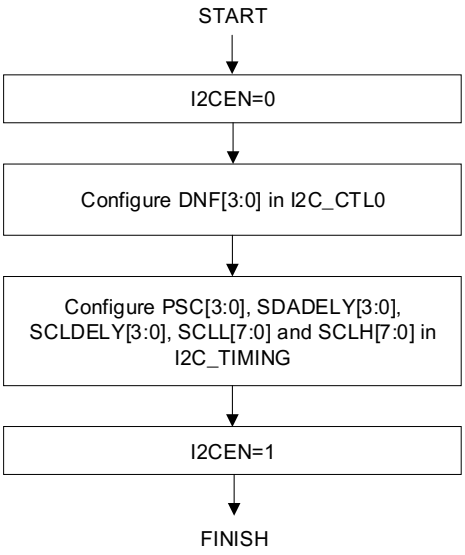
sequence must be executed, or only the header to be sent. The number of bytes to be transferred should be configured in BYTENUM[7:0] in I2C_CTL1 register. If the number of bytes to be transferred is equal to or greater than 255, BYTENUM[7:0] should be configured as 0xFF. Then the master sends the START 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 24-30. 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 signal to 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 24-31. Programming model for master transmitting (N<=255)

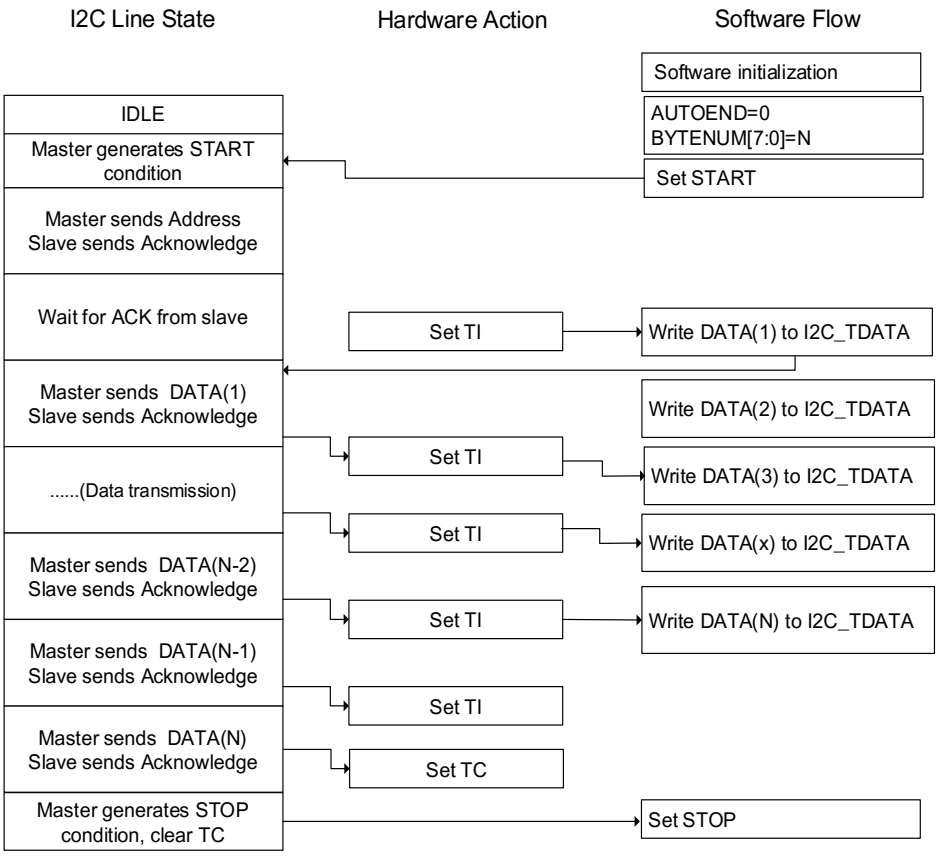
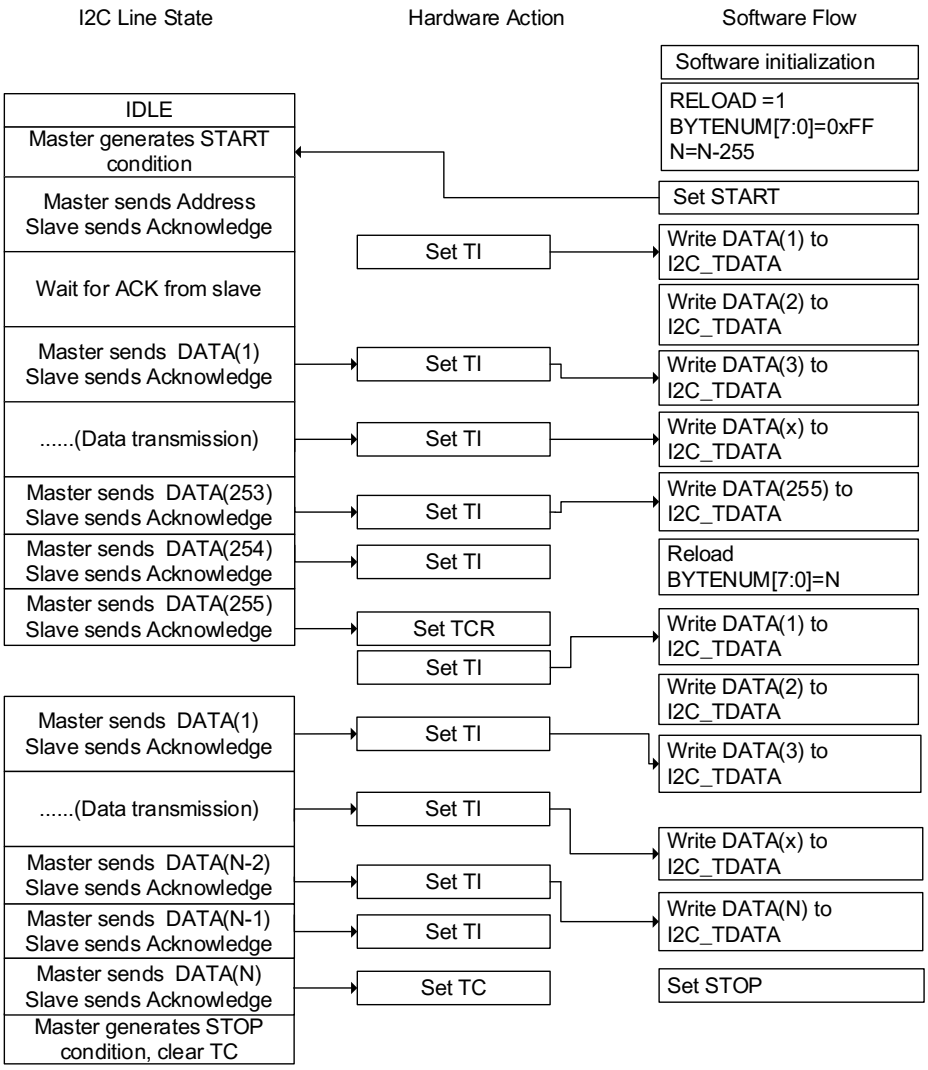


Figure 24-32. Programming model for master transmitting (N>255)



■ **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 signal by 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 24-33. Programming model for master receiving (N<=255)

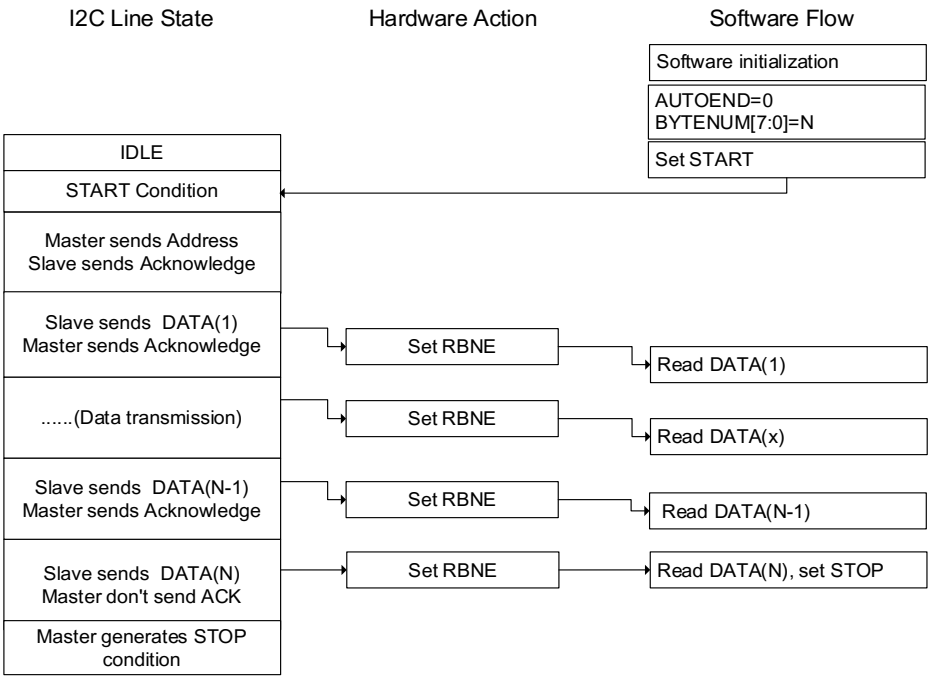
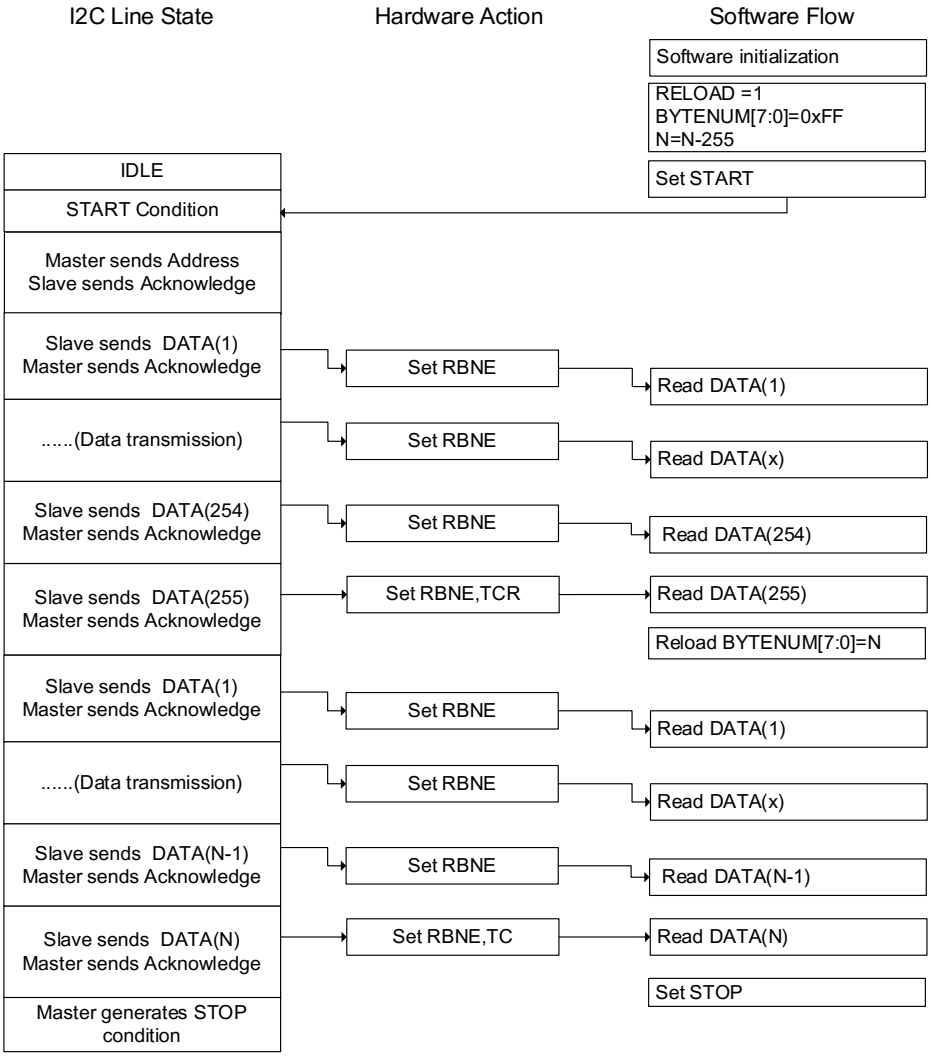


Figure 24-34. Programming model for master receiving (N>255)



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. The Slave Byte Control mode must be 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 interval 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.

SMBus mode

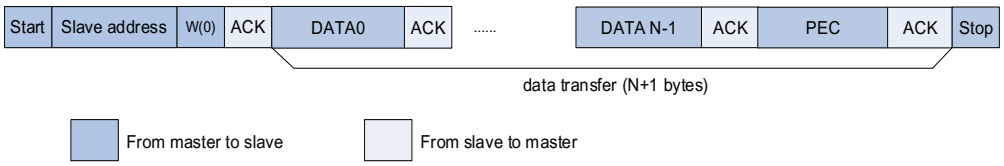
- **SMBus master transmitter and slave receiver**

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. 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: After the RELOAD bit is set, the PECTRANS cannot be changed.

Figure 24-35. 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 signal 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 signal.

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.

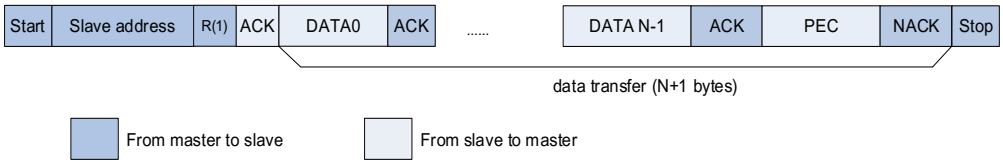
When used as slave transmitter, in order to allow PEC transmission at the end of BYTENUM[7:0] bytes, SBCTL must be set. If PECTRANS bit is set, the number of bytes in BYTENUM[7:0] contains PEC byte. In this case, if the number of bytes requested by the master is greater than BYTENUM-1, the total number of TI interrupts will be BYTENUM-1, and the data of the I2C_PEC register will be transmitted automatically.

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[7:0] must be configured before detecting the START signal.

Figure 24-36. SMBus master receiver and slave transmitter communication flow



Wakeup from power saving modes

When the address of I2C matches correctly, it can wake up MCU from sleep mode and deep-sleep mode. In order to wake up from these power saving modes, WUEN bit must be set in the I2C_CTL0 register and the IRC16M must be selected as the clock source for I2CCLK. During deep-sleep mode, the IRC16M is switched off. The I2C interface switches the IRC16M on, and stretches SCL low until IRC16M is woken up when a START is detected. Then the IRC16M 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, IRC16M 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, IRC16M 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, the I2C peripheral

must be disabled (I2CEN=0) if wakeup from power saving mode is disabled (WUEN =0).

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.

I2C error and interrupts

The I2C error flags are listed in *Table 24-7. I2C error flags*.

Table 24-7. 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 24-8. I2C interrupt events*.

Table 24-8. I2C interrupt events

Interrupt event	Event flag	Enable control bit
I2C_RDATA is not empty during receiving	RBNE	RBNEIE
Transmit interrupt	TI	TIE
STOP signal detected in slave mode	STPDET	STPDETIE
Transfer complete reload	TCR	TCIE
Transfer complete	TC	
Address match	ADDSEND	ADDMIE
Not acknowledge received	NACK	NACKIE
Bus error	BERR	ERRIE
Arbitration Lost	LOSTARB	
Overrun / Underrun error	OUERR	
PEC error	PECERR	
Timeout error	TIMEOUT	

Interrupt event	Event flag	Enable control bit
SMBus Alert	SMBALT	

I2C debug mode

When the microcontroller enters the debug mode (Cortex®-M33 core halted), the SMBus timeout either continues to work normally or stops, depending on the I2Cx_HOLD configuration bits in the DBG module.

24.2.4. Register definition

I2C3 base address: 0x4000 8000

I2C4 base address: 0x4000 8400

I2C5 base address: 0x4000 8800

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 0: PEC Calculation off 1: PEC Calculation on
22	SMBALTEN	SMBus Alert enable 0: SMBA pin is not pulled down (device mode) or SMBus Alert pin SMBA is disabled (host mode) 1: SMBA pin is pulled down (device mode) or SMBus Alert pin SMBA is enabled (host mode)
21	SMBDAEN	SMBus device default address enable 0: Device default address is disabled, the default address 0b1100001x will be not acknowledged. 1: Device default address is enabled, the default address 0b1100001x will be acknowledged.
20	SMBHAEN	SMBus Host address enable 0: Host address is disabled, address 0b0001000x will be not acknowledged. 1: Host address is enabled, address 0b0001000x will be acknowledged.
19	GCEN	Whether or not to response to a General Call (0x00) 0: Slave won't response to a General Call

		1: Slave will response to a General Call
18	WUEN	Wakeup from power saving mode enable This bit is cleared when mcu wakeup from power saving mode. 0: Wakeup from power saving mode disable. 1: Wakeup from power saving mode enable. **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. This bit is set and cleared by software. 0: SCL Stretching is enabled 1: SCL Stretching is disabled **Note:** When in master mode, this bit must be 0. This bit can be modified when I2CEN = 0.
16	SBCTL	Slave byte control This bit is used to enable hardware byte control in slave mode. 0: Slave byte control is disabled 1: Slave byte control is enabled
15	DENR	DMA enable for reception 0: DMA is disabled for reception 1: DMA is enabled for reception
14	DENT	DMA enable for transmission 0: DMA is disabled for transmission 1: DMA is enabled for transmission
13	Reserved	Must be kept at reset value.
12	ANOFF	Analog noise filter disable 0: Analog noise filter is enabled 1: Analog noise filter is disabled **Note:** This bit can only be programmed when the I2C is disabled (I2CEN = 0).
11:8	DNF[3:0]	Digital noise filter 0000: Digital filter is disabled 0001: Digital filter is enabled and filter spikes with a length of up to 1 t_{I2CCLK} ... 1111: Digital filter is enabled and filter spikes with a length of up to15 t_{I2CCLK} These bits can only be modified when the I2C is disabled (I2CEN = 0).
7	ERRIE	Error interrupt enable 0: Error interrupt disabled 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 0: Transfer complete interrupt is disabled

		1: Transfer complete interrupt is enabled
5	STPDETIE	Stop detection interrupt enable 0: Stop detection (STPDET) interrupt is disabled 1: Stop detection (STPDET) interrupt is enabled
4	NACKIE	Not acknowledge received interrupt enable 0: Not acknowledge (NACK) received interrupt is disabled 1: Not acknowledge (NACK) received interrupt is enabled
3	ADDMIE	Address match interrupt enable in slave mode 0: Address match interrupt is disabled 1: Address match interrupt is enabled
2	RBNEIE	Receive interrupt enable 0: Receive (RBNE) interrupt is disabled 1: Receive (RBNE) interrupt is enabled
1	TIE	Transmit interrupt enable 0: Transmit (TI) interrupt is disabled 1: Transmit (TI) interrupt is enabled
0	I2CEN	I2C peripheral enable 0: I2C is disabled 1: I2C is enabled

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 Set by software. Cleared by hardware in the following cases: When PEC byte is transferred or ADDSEND bit is set or STOP signal is detected or

		I2CEN=0. 0: Don't transfer PEC value 1: Transfer PEC **Note:** This bit has no effect when RELOAD=1, or SBCTL=0 in slave mode.
25	AUTOEND	Automatic end mode in master mode 0: TC bit is set when the transfer of BYTENUM[7:0] bytes is completed. 1: a STOP signal is sent automatically when the transfer of BYTENUM[7:0] bytes is completed. **Note:** This bit works only when RELOAD=0. This bit is set and cleared by software.
24	RELOAD	Reload mode 0: After the data of BYTENUM[7:0] bytes transfer, the transfer is completed. 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. This bit is set and cleared by software.
23:16	BYTENUM[7:0]	Number of bytes to be transferred These bits are programmed with the number of bytes to be transferred. When SBCTL=0, these bits have no effect. **Note:** These bits should not be modified when the START bit is set, and the software cannot change these bits during transmission.
15	NACKEN	Generate NACK in slave mode 0: an ACK is sent after receiving a new byte. 1: a NACK is sent after receiving a new byte. **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 This bit is set by software and cleared by hardware when I2CEN=0 or STOP condition is detected. 0: STOP will not be sent 1: STOP will be sent
13	START	Generate a START condition on I2C bus 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. 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

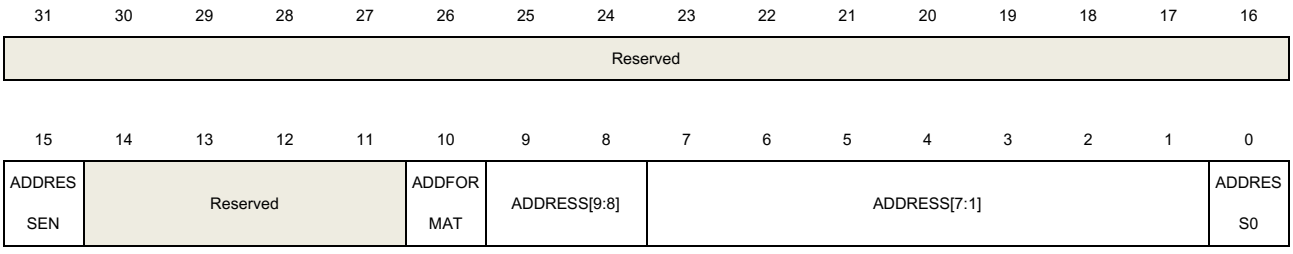
		0: The 10 bit master receive address sequence is START + header of 10-bit address (write) + slave address byte 2 + RESTART + header of 10-bit address (read). 1: The 10 bit master receive address sequence is RESTART + header of 10-bit address (read). **Note:** When the START bit is set, this bit can not be changed, and the software cannot change this bit during transmission.
11	ADD10EN	10-bit addressing mode enable in master mode 0: 7-bit addressing in master mode 1: 10-bit addressing in master mode **Note:** When the START bit is set, this bit can not be modified, and the software cannot change this bit during transmission.
10	TRDIR	Transfer direction in master mode 0: Master transmit 1: Master receive **Note:** When the START bit is set, this bit can not be modified, and the software cannot change this bit during transmission.
9:0	SADDRESS[9:0]	Slave address to be sent SADDRESS[9:8]: Slave address bit 9:8 If ADD10EN = 0, these bits have no effect. If ADD10EN = 1, these bits should be written with bits 9:8 of the slave address to be sent. SADDRESS[7:1]: Slave address bit 7:1 If ADD10EN = 0, these bits should be written with the 7-bit slave address to be sent. If ADD10EN = 1, these bits should be written with bits 7:1 of the slave address to be sent. SADDRESS0: Slave address bit 0 If ADD10EN = 0, this bit has no effect. If ADD10EN = 1, this bit should be written with bit 0 of the slave address to be sent **Note:** When the START bit is set, the bit filed can not be modified, and the software cannot change these bits during transmission.

Slave address register 0 (I2C_SADDR0)

Address offset: 0x08

Reset value: 0x0000 0000

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



rw rw rw rw rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	ADDRESSEN	I2C address enable 0: I2C address disable. 1: I2C address enable.
14:11	Reserved	Must be kept at reset value.
10	ADDFORMAT	Address mode for the I2C slave 0: 7-bit address 1: 10-bit address **Note:** When ADDRESSEN is set, this bit should not be written.
9:8	ADDRESS[9:8]	Highest two bits of a 10-bit address **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 **Note:** 1. When ADDRESSEN is set, this bit should not be written. 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 **Note:** When ADDRESSEN is set, this bit should not be written.

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. 1: Second I2C address enable.

14:11	Reserved	Must be kept at reset value.
10:8	ADDMSK2[2:0]	ADDRESS2[7:1] mask Defines which bits of ADDRESS2[7:1] are compared with an incoming address byte, and which bits are masked (don't care). 000: No mask, all the bits must be compared. n (001~110): ADDRESS2[n:1] is masked. Only ADDRESS2[7:n+1] are compared. 111: ADDRESS2[7:1] are masked. All 7-bit received addresses are acknowledged except the reserved address (0b0000xxx and 0b1111xxx). **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 **Note:** 1. When ADDRESS2EN is set, these bits should not be written. 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.

Timing register (I2C_TIMING)

Address offset: 0x10
Reset value: 0x0000 0000

This register cannot be modified when the I2C is enabled (I2CEN = 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
PSC[3:0]				Reserved				SCLDELY[3:0]				SDADELY[3:0]			
rw								rw				rw			

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

Bits	Fields	Descriptions
31:28	PSC[3:0]	Timing prescaler 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. $t_{PSC}=(PSC+1)*t_{I2CCLK}$
27:24	Reserved	Must be kept at reset value.
23:20	SCLDELY[3:0]	Data setup time 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

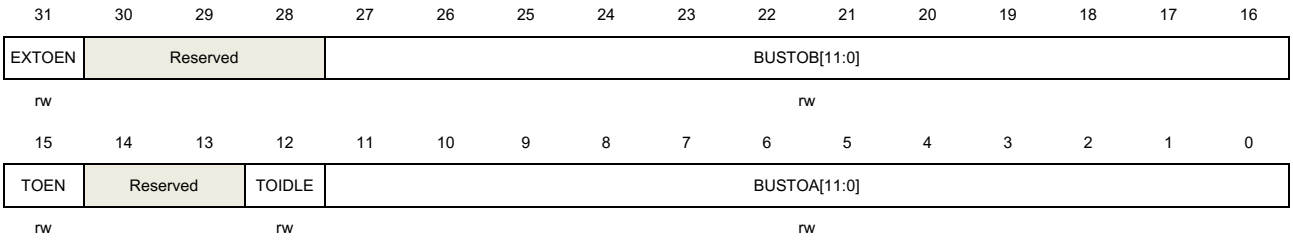
		mode and in slave mode when SS = 0. $t_{SCLDELY}=(SCLDELY+1)*t_{PSC}$
19:16	SDADELY[3:0]	Data hold time A delay $t_{SDADELY}$ between SCL falling edge and SDA edge can be generated by configuring these bits. And during $t_{SDADELY}$, the SCL line is stretched low in master mode and in slave mode when SS = 0. 0x0: $t_{SDADELY}$= 1 * t_{PSC} 0x1-0xF: $t_{SDADELY}$= SDADELY * t_{PSC}
15:8	SCLH[7:0]	SCL high period SCL high period can be generated by configuring these bits. $t_{SCLH}=(SCLH+1)*t_{PSC}$ **Note:** These bits can only be used in master mode.
7:0	SCLL[7:0]	SCL low period SCL low period can be generated by configuring these bits. $t_{SCLL}=(SCLL+1)*t_{PSC}$ **Note:** These bits can only be used in master mode.

Timeout register (I2C_TIMEOUT)

Address offset: 0x14

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	EXTOEN	Extended clock timeout detection enable 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}. 0: Extended clock timeout detection is disabled. 1: Extended clock timeout detection is enabled.
30:28	Reserved	Must be kept at reset value.
27:16	BUSTOB[11:0]	Bus timeout B Configure the cumulative clock extension timeout. In master mode, the master cumulative clock low extend time $t_{LOW:MEXT}$ is detected. In slave mode, the slave cumulative clock low extend time $t_{LOW:SEXT}$ is detected. $t_{LOW:EXT}=(BUSTOB+1)*2048*t_{I2CCLK}$.

		Note: These bits can be modified only when EXTOEN =0.
15	TOEN	Clock timeout detection enable 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. 0: SCL timeout detection is disabled 1: SCL timeout detection is enabled
14:13	Reserved	Must be kept at reset value.
12	TOIDLE	Idle clock timeout detection 0: BUSTOA is used to detect SCL low timeout 1: BUSTOA is used to detect both SCL and SDA high timeout when the bus is idle **Note:** This bit can be written only when TOEN =0.
11:0	BUSTOA[11:0]	Bus timeout A When TOIDLE = 0, $t_{TIMEOUT}$=(BUSTOA+1)*2048*t_{I2CCLK}. When TOIDLE = 1, t_{IDLE}=(BUSTOA+1)*4*t_{I2CCLK}. **Note:** These bits can be written only when TOEN =0.

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 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. 0: Receiver 1: Transmitter
15	I2CBSY	Busy flag

		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. 0: No I2C communication. 1: I2C communication active.
14	Reserved	Must be kept at reset value.
13	SMBALT	SMBus Alert 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. 0: SMBALERT event is not detected on SMBA pin 1: SMBALERT event is detected on SMBA pin
12	TIMEOUT	TIMEOUT flag. 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. 0: no timeout or extended clock timeout occur 1: a timeout or extended clock timeout occur
11	PECERR	PEC error 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. 0: Received PEC and content of I2C_PEC match 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 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. 0: No overrun or underrun occurs 1: Overrun or underrun occurs
9	LOSTARB	Arbitration Lost It is cleared by software by setting the LOSTARBC bit and cleared by hardware when I2CEN=0. 0: No arbitration lost. 1: Arbitration lost occurs and the I2C block changes back to slave mode.
8	BERR	Bus error 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. 0: No bus error 1: A bus error detected

7	TCR	Transfer complete reload 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. 0: When RELOAD=1, transfer of BYTENUM[7:0] bytes is not completed 1: When RELOAD=1, transfer of BYTENUM[7:0] bytes is completed
6	TC	Transfer complete in master mode 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. 0: Transfer of BYTENUM[7:0] bytes is not completed 1: Transfer of BYTENUM[7:0] bytes is completed
5	STPDET	STOP signal detected 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. 0: STOP signal is not detected. 1: STOP signal is detected.
4	NACK	Not Acknowledge flag 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. 0: ACK is received. 1: NACK is received.
3	ADDSEND	Address received matches its own address in slave mode. 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. 0: Received address not matched 1: Received address matched
2	RBNE	I2C_RDATA is not empty during receiving 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. 0: I2C_RDATA is empty 1: I2C_RDATA is not empty, software can read
1	TI	Transmit interrupt 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). 0: I2C_TDATA is not empty or the I2C is not ready to transmit data 1: I2C_TDATA is empty and the I2C is ready to transmit data
0	TBE	I2C_TDATA is empty during transmitting

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.
0: I2C_TDATA is not empty
1: I2C_TDATA is empty

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. Software can clear the SMBALT bit of I2C_STAT by writing 1 to this bit.
12	TIMEOUTC	TIMEOUT flag clear. Software can clear the TIMEOUT bit of I2C_STAT by writing 1 to this bit.
11	PECERRC	PEC error flag clear. Software can clear the PECERR bit of I2C_STAT by writing 1 to this bit.
10	OUERRC	Overrun/Underrun flag clear. Software can clear the OUERR bit of I2C_STAT by writing 1 to this bit.
9	LOSTARBC	Arbitration Lost flag clear. Software can clear the LOSTARB bit of I2C_STAT by writing 1 to this bit.
8	BERRC	Bus error flag clear. Software can clear the BERR bit of I2C_STAT by writing 1 to this bit.
7:6	Reserved	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 Software can clear the NACK bit of I2C_STAT by writing 1 to this bit.
3	ADDSENDC	ADDSEND flag clear

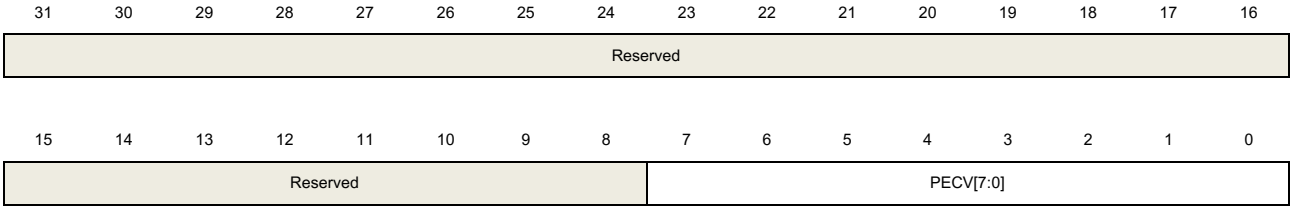
		Software can clear the ADDSEND bit of I2C_STAT by writing 1 to this bit.
2:0	Reserved	Must be kept at reset value.

PEC register (I2C_PEC)

Address offset: 0x20

Reset value: 0x0000 0000

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



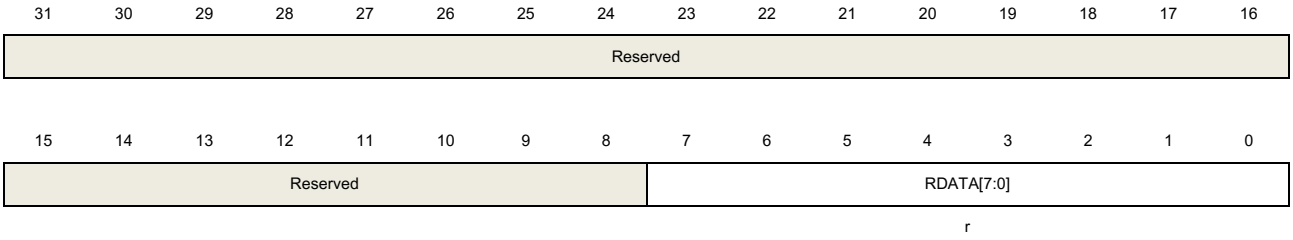
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	PECV[7:0]	Packet Error Checking Value that calculated by hardware when PEC is enabled. PECV is cleared by hardware when I2CEN = 0.

Receive data register (I2C_RDATA)

Address offset: 0x24

Reset value: 0x0000 0000

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



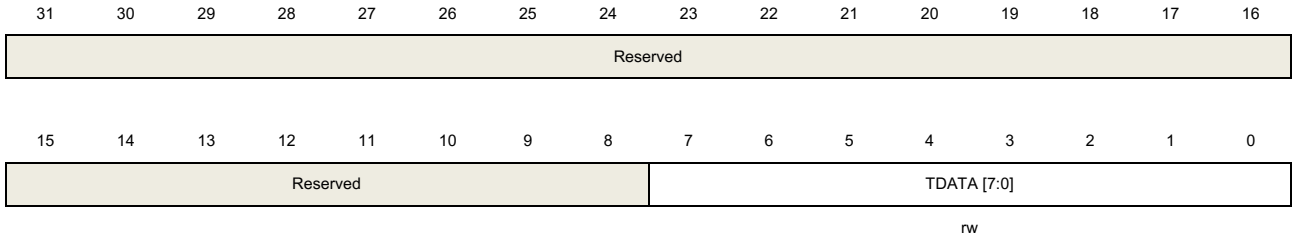
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	RDATA[7:0]	Receive data value

Transmit data register (I2C_TDATA)

Address offset: 0x28

Reset value: 0x0000 0000

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



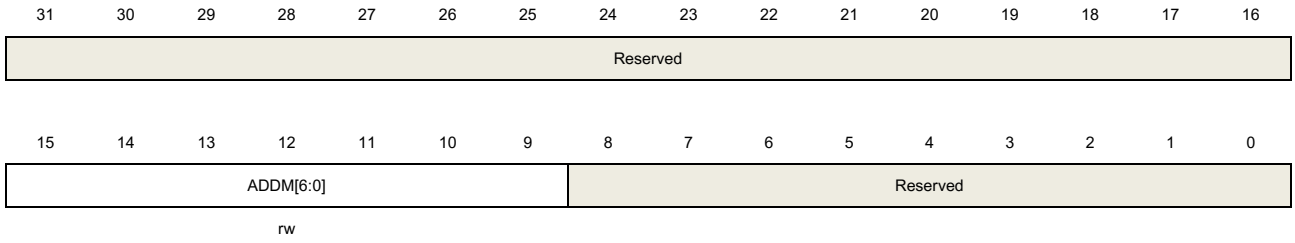
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	TDATA[7:0]	Transmit data value

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.

25. Serial peripheral interface/Inter-IC sound (SPI/I2S)

25.1. 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 and simplex communication modes are supported, with hardware CRC calculation and checking. Quad-SPI master mode is also supported in SPI5.

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. (By using two extra I2S modules called I2S1_ADD and I2S2_ADD, I2S full duplex mode is also supported in I2S1 and I2S2.)

25.2. Characteristics

25.2.1. SPI characteristics

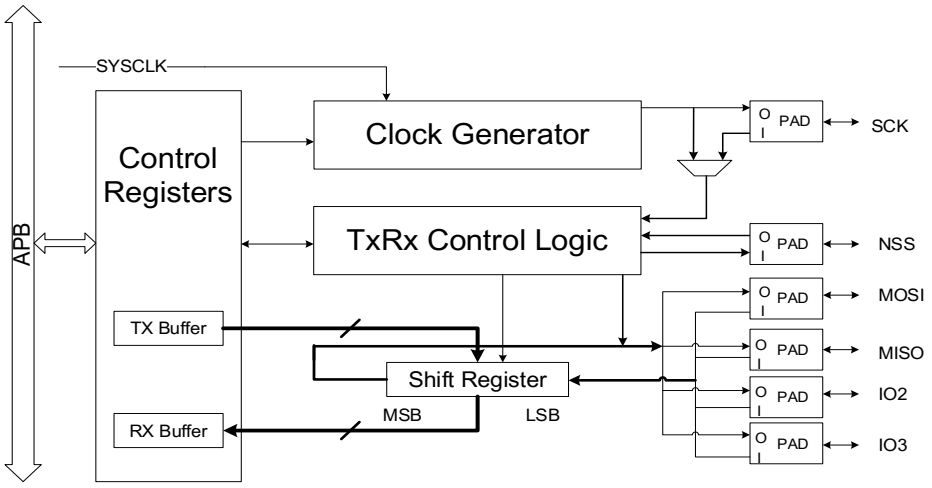
- Master or slave operation with full-duplex or simplex mode
- Separate transmit and receive buffer, 16 bits wide
- Data frame size can be 8 or 16 bits
- Bit order can be LSB first or MSB first
- Software and hardware NSS management
- Hardware CRC calculation , transmission and checking
- Transmission and reception using DMA
- SPI TI mode supported
- Quad-SPI configuration available in master mode(only in SPI5)

25.2.2. I2S characteristics

- Master or slave operation with transmission or reception mode
- Master or slave operation with full-duplex mode(only in SPI1 and SPI2)
- 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
- Transmission and reception using a 16 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 using DMA

25.3. SPI block diagram

Figure 25-1. Block diagram of SPI



25.4. SPI signal description

25.4.1. Normal configuration (Not Quad-SPI Mode)

Table 25-1. SPI signal description

Pin Name	Direction	Description
SCK	I/O	Master: SPI Clock Output Slave: SPI Clock Input
MISO	I/O	Master: Data reception line Slave: Data transmission line Master with Bidirectional mode: Not used Slave with Bidirectional mode: Data transmission and reception Line.
MOSI	I/O	Master: Data transmission line Slave: Data reception line Master with Bidirectional mode: Data transmission and reception Line. Slave with Bidirectional mode: Not used
NSS	I/O	Software NSS mode: not used 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. Slave in hardware NSS mode: NSS input, as a chip select

		signal for slave.

25.4.2. 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 (only available in SPI5). Quad-SPI mode can only work at master mode.

The SPI is connected to external devices through 6 pins in Quad-SPI mode:

Table 25-2. 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

25.5. SPI function overview

25.5.1. SPI clock timing and data format

CKPL and CKPH bits in SPI_CTL0 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 25-2. SPI timing diagram in normal mode

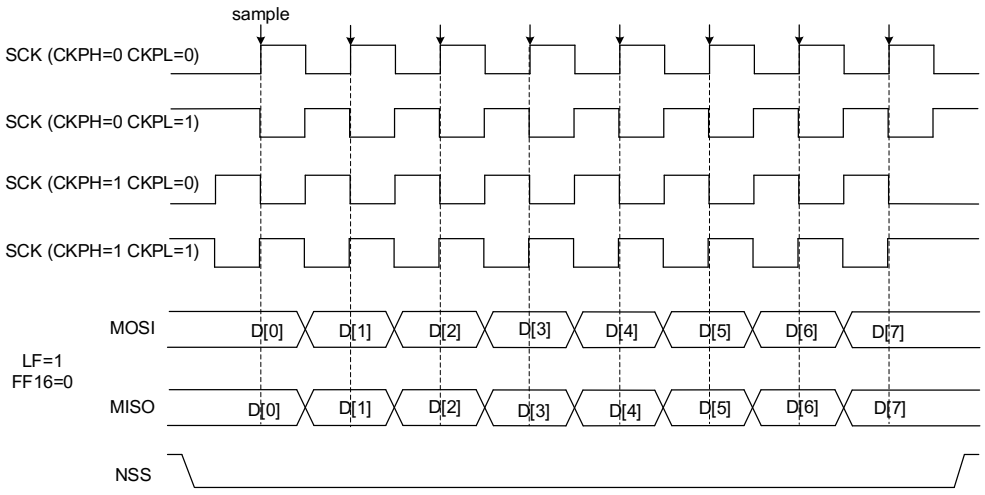
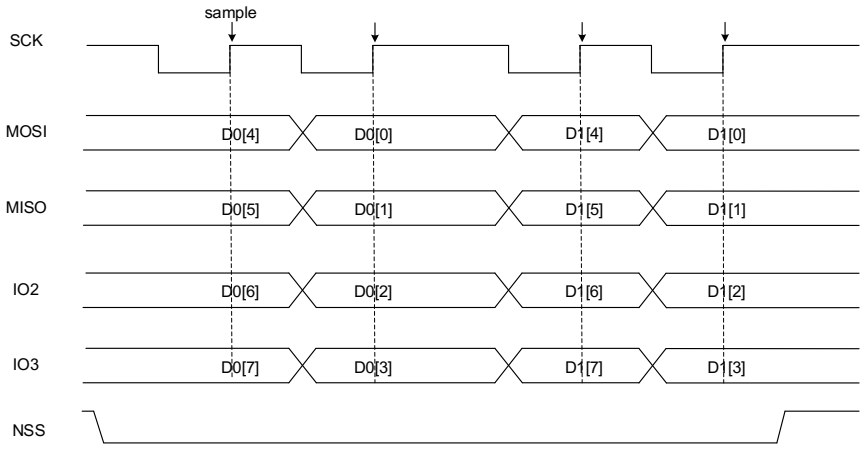


Figure 25-3. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0)



In normal mode, the length of data is configured by the FF16 bit in the SPI_CTL0 register. Data length is 16 bits if FF16=1, otherwise is 8 bits. The data frame length is fixed to 8 bits in Quad-SPI mode.

Data order is configured by LF bit in SPI_CTL0 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.

25.5.2. NSS function

Slave Mode

When slave mode is configured (MSTMOD = 0), SPI gets NSS level from NSS pin in hardware NSS mode (SWNSSEN = 0) or from SWNSS bit in software NSS mode (SWNSSEN = 1) and transmits/receives data only when NSS level is low. In software NSS mode, NSS pin is not used.

Table 25-3. NSS function in slave mod

Mode	Register configuration	Description
Slave hardware NSS mode	MSTMOD = 0 SWNSSEN = 0	SPI slave gets NSS level from NSS pin.
Slave software NSS mode	MSTMOD = 0 SWNSSEN = 1	SPI slave NSS level is determined by the SWNSS bit. SWNSS = 0: NSS level is low SWNSS = 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 (SWNSSEN=0, NSSDRV=0) or software mode (SWNSSEN=1). Then, once the NSS pin (in hardware NSS mode) or the SWNSS bit (in software NSS mode) goes low, the SPI automatically enters to 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 (SWNSSEN=0, NSSDRV=1). NSS goes low after SPI is enabled and goes low when transmission or reception process begins.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

Table 25-4. NSS function in master mod

Mode	Register configuration	Description
Master hardware NSS output mode	MSTMOD = 1 SWNSSEN = 0 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 goes low after enabling SPI.
Master hardware NSS input mode	MSTMOD = 1 SWNSSEN = 0 NSSDRV=0	Applicable to multi-master mode. At this time, NSS is configured as hardware input mode. Once the NSS pin is pulled low, 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 SWNSSEN = 1 SWNSS = 0 NSSDRV: Don't care	Applicable to multi-master mode. Once SWNSS = 0, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be 1.
	MSTMOD = 1 SWNSSEN = 1 SWNSS = 1 NSSDRV: Don't care	The slave can use hardware or software NSS mode.

25.5.3. SPI operation modes

Table 25-5. SPI operation modes

Mode	Description	Register Configuration	Data Pin Usage
MFD	Master Full-Duplex	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Reception
MTU	Master Transmission with unidirectional connection	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Not used
MRU	Master Reception with	MSTMOD = 1	MOSI: Not used

Mode	Description	Register Configuration	Data Pin Usage
	unidirectional connection	RO = 1 BDEN = 0 BDOEN: Don't care	MISO: Reception
MTB	Master Transmission with bidirectional connection	MSTMOD = 1 RO = 0 BDEN = 1 BDOEN = 1	MOSI: Transmission MISO: Not used
MRB	Master Reception with bidirectional connection	MSTMOD = 1 RO = 0 BDEN = 1 BDOEN = 0	MOSI: Reception MISO: Not used
SFD	Slave Full-Duplex	MSTMOD = 0 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI : Reception MISO: Transmission
STU	Slave Transmission with unidirectional connection	MSTMOD = 0 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Not used MISO: Transmission
SRU	Slave Reception with unidirectional connection	MSTMOD = 0 RO = 1 BDEN = 0 BDOEN: Don't care	MOSI: Reception MISO: Not used
STB	Slave Transmission with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 1	MOSI: Not used MISO: Transmission
SRB	Slave Reception with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 0	MOSI: Not used MISO: Reception

Figure 25-4. A typical Full-duplex connection

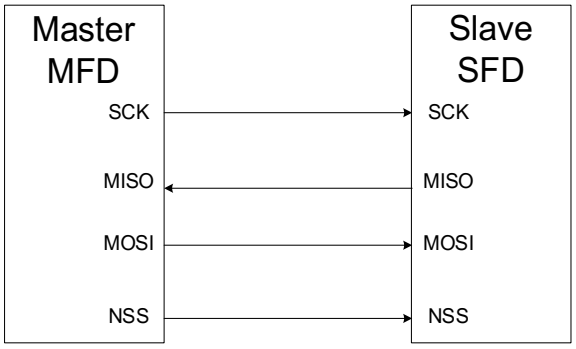


Figure 25-5. A typical simplex connection (Master: Receive, Slave: Transmit)

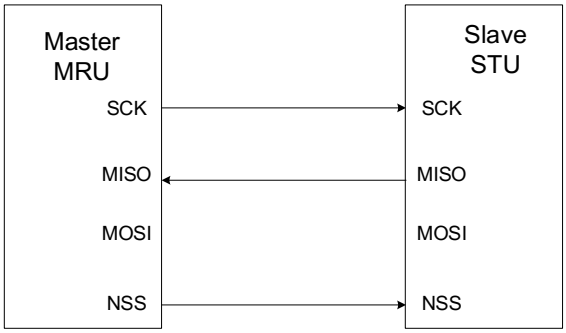


Figure 25-6. A typical simplex connection (Master: Transmit only, Slave: Receive)

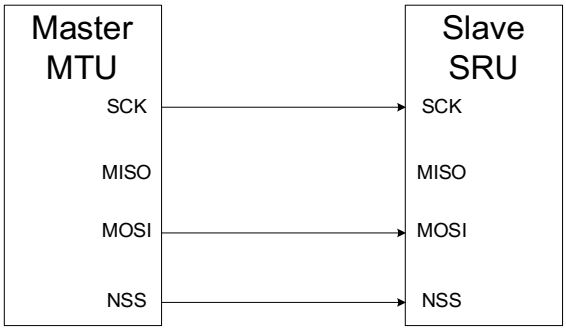
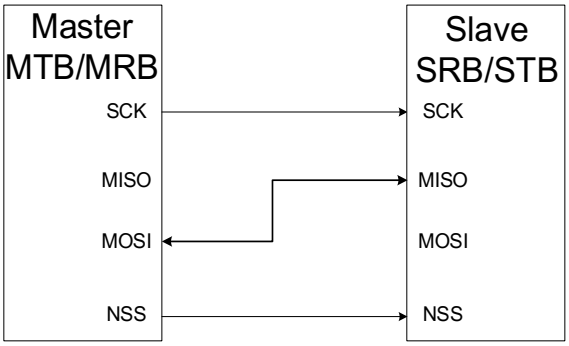


Figure 25-7. A typical bidirectional connection



SPI initialization sequence

Before transmiting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode or slave TI mode is used, program the PSC[2:0] bits in SPI_CTL0 register to generate SCK with desired baud rate or configure the Td time in TI mode, otherwise, ignore this step.
2. Program data format (FF16 bit in the SPI_CTL0 register).
3. Program the clock timing register (CKPL and CKPH bits in the SPI_CTL0 register).
4. Program the frame format (LF bit in the SPI_CTL0 register).
5. Program the NSS mode (SWNSSEN and NSSDRV bits in the SPI_CTL0 register) according to the application's demand as described above in ***NSS function*** section.

6. If TI mode is used, set TMOD bit in SPI_CTL1 register, otherwise, ignore this step.
7. Configure MSTMOD, RO, BDEN and BDOEN depending on the operation modes described in *Table 25-5. SPI operation modes*.
8. If Quad-SPI mode is used, set the QMOD bit in SPI_QCTL register. Ignore this step if Quad-SPI mode is not used.
9. Enable the SPI (set the SPIEN bit).

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 transmit buffer. In slave mode the transmission starts when SCK clock signal begins to toggle at SCK pin and NSS level is low, so application should ensure that data is already written into transmit buffer before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the data buffer to the shift register and then begins to transmit the loaded data frame, TBE (transmit buffer empty) flag is set after the first bit of this frame is transmited. After TBE flag is set, which means the transmit buffer is empty, the application should write SPI_DATA register again if it has more data to transmit.

In master mode, software should write the next data into SPI_DATA register before the transmission of current data frame is completed if it desires to generate continuous transmission.

Reception sequence

The incoming data will be moved from shift register to the receive buffer after the last valid sample clock and also, RBNE (receive buffer not empty) will be set. The application should read SPI_DATA register to get the received data and this will clear the RBNE flag automatically. 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 transmit buffer is not empty.

SPI operation sequence in different modes (Not Quad-SPI or TI mode)

In full-duplex mode, either MFD or SFD, application should monitor the RBNE and TBE flags and follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to full-duplex mode, except that application should ignore the RBNE and OVRE flags and only perform transmission sequence described above.

In master reception mode (MRU or MRB), the behavior is different from full-duplex mode or transmission mode. In MRU or MRB mode, the SPI continuously generates SCK just after

SPI is enabled, until the SPI is disabled. So the application should ignore the TBE flag and read out reception buffer in time after the RBNE flag is set, otherwise a data overrun fault will occur.

The slave reception mode (SRU or SRB) is similar to full-duplex mode, except that application should ignore the TBE flag and only perform reception sequence described above.

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 and CKPH bits in SPI_CTL0 registers take no effect and the SCK sample edge is falling edge.

Figure 25-8. Timing diagram of TI master mode with discontinuous transfer

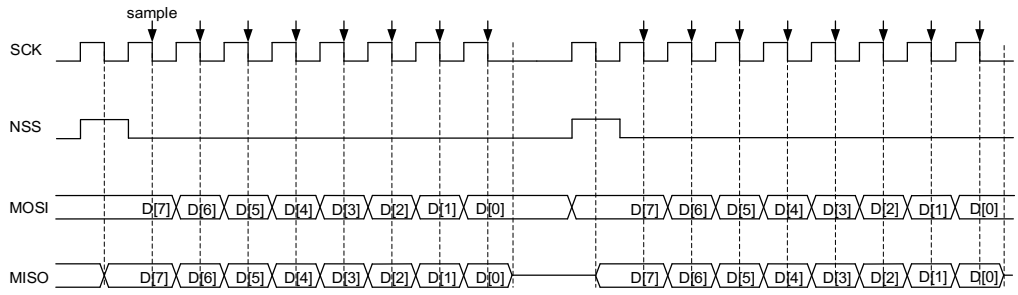
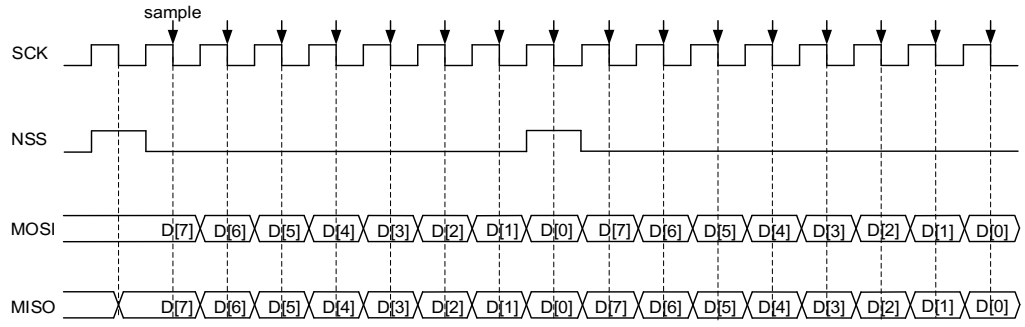
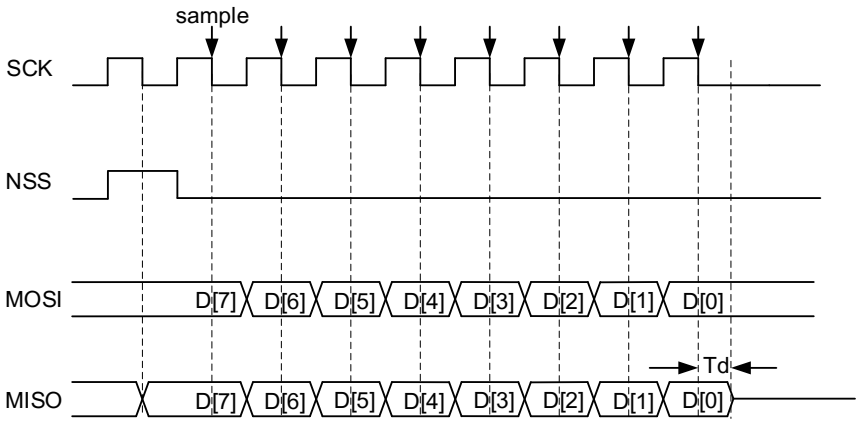


Figure 25-9. Timing diagram of TI master mode with continuous transfer



In master TI mode, SPI can perform continuous or non-continuous transfer. If the master writes SPI_DATA 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 25-10. 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 pin. This time is called T_d. T_d is decided by PSC[2:0] bits in SPI_CTL0 register.

$$T_d = \frac{T_{bit}}{2} + 5 * T_{pclk} \quad (25\text{-}1)$$

For example, if PSC[2:0] = 010, T_d is 9*Tpclk.

In slave mode, the slave also monitors the NSS signal and sets an error flag FE 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 TBE bit is set and TRANS bit is cleared, then set QMOD bit in SPI_QCTL register. In Quad-SPI mode, BDEN, BDOEN, CRCEN, CRCNT, FF16, RO and LF in SPI_CTL0 register should be kept cleared and MSTMOD should be set to ensure that SPI is in master mode. SPIEN, PSC, CKPL and CKPH should be configured as desired.

There are 2 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_DATA (TBE is cleared) and SPIEN is set. Once SPI starts transmission, it always checks TBE status at the end of a frame and stops when condition is not met.

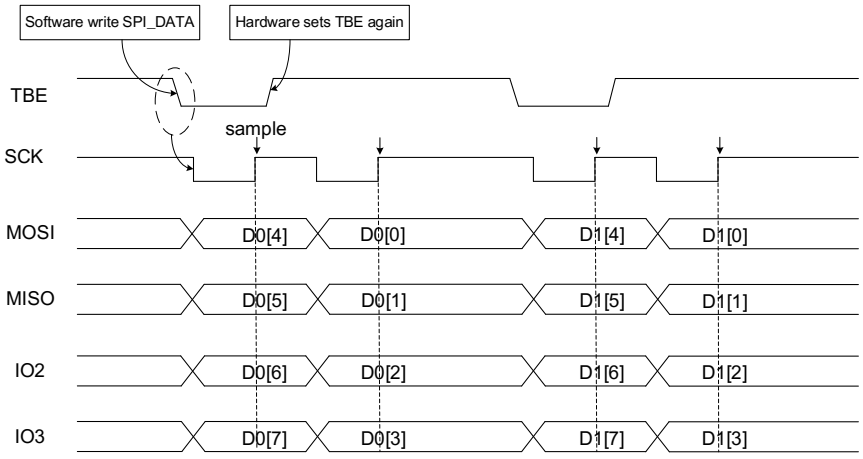
The operation flow for transmitting in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 and SPI_CTL1 based

on your application requirements.

2. Set QMOD bit in SPI_QCTL register and then enable SPI by setting SPIEN in SPI_CTL0.
3. Write the byte to SPI_DATA register and the TBE will be cleared.
4. Wait until TBE is set by hardware again before writing the next byte.

Figure 25-11. Timing diagram of quad write operation in Quad-SPI mode



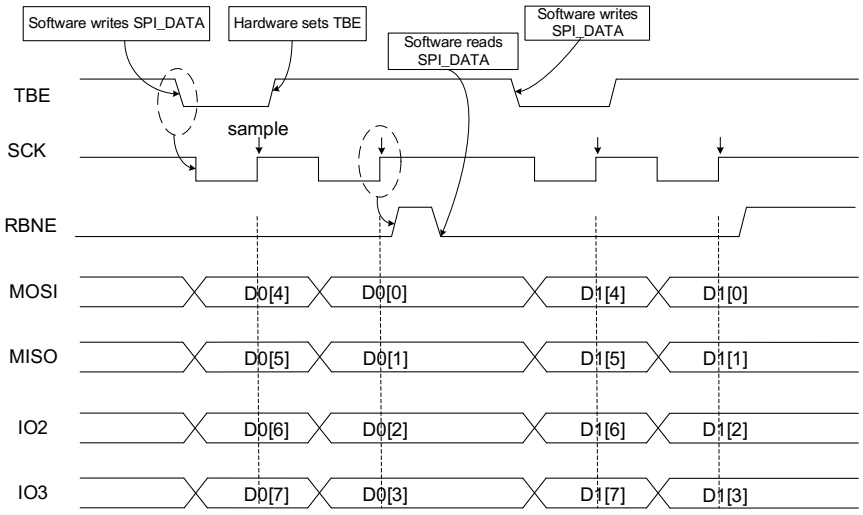
Quad read operation

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_DATA (TBE is cleared) and SPIEN is set. Writing data into SPI_DATA is only to generate SCK clocks, so the written data can be any value. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition is not met. So, software should always write dummy data into SPI_DATA to make SPI generate SCK.

The operation flow for receiving in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI_CTL0 and SPI_CTL1 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_DATA register.
4. Wait until the RBNE flag is set and read SPI_DATA to get the received byte.
5. Write an arbitrary byte (for example, 0xFF) to SPI_DATA to receive the next byte.

Figure 25-12. 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 SFD

Wait for the last RBNE flag and then receive the last data. Confirm that TBE=1 and TRANS=0. At last, disable the SPI by clearing SPIEN bit.

MTU MTB STU STB

Write the last data into SPI_DATA and wait until the TBE flag is set and then wait until the TRANS flag is cleared. Disable the SPI by clearing SPIEN bit.

MRU MRB

After getting the second last RBNE flag, read out this data and delay for a SCK clock time and then, disable the SPI by clearing SPIEN bit. Wait until the last RBNE flag is set and read out the last data.

SRU SRB

Application can disable the SPI when it doesn't want to receive data, and then wait until the TRANS=0 to ensure the on-going transfer completes.

TI mode

The disabling sequence of TI mode is the same as the sequences described above.

Quad-SPI mode

Before leaving quad wire mode or disabling SPI, software should first check that, TBE bit is set and TRANS bit is cleared, then the QMOD bit in SPI_QCTL register and SPIEN bit in SPI_CTL0 register are cleared.

25.5.4. DMA function

The DMA function frees the application from data writing and reading process during transfer, thus improve the system efficiency.

DMA function in SPI is enabled by setting DMATEN and DMAREN bits in SPI_CTL1 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 being enabled, If DMATEN is set, SPI will generate a DMA request each time TBE=1, then DMA will acknowledge to this request and write data into the SPI_DATA register automatically. If DMAREN is set, SPI will generate a DMA request each time RBNE=1, then DMA will acknowledge to this request and read data from the SPI_DATA register automatically.

25.5.5. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial in SPI_CRCPOLY register.

Application can switch on the CRC function by setting CRCEN bit in SPI_CTL0 register. 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 register.

To transmit the calculated CRC value, application should set the CRCNT bit in SPI_CTL0 register after the last data is written to the transmit buffer. In full-duplex mode (MFD or SFD) the SPI treats the incoming data as a CRC value when it transmits a CRC and will check the received CRC value. In reception mode (MRB, MRU, SRU and SRB), the application should set the CRCNT bit after the second-last data frame is received. When CRC checking fails, the CRCERR flag will be set.

If DMA function is enabled, application doesn't need to operate CRCNT bit and hardware will automatically process the CRC transmitting and checking.

Note: When SPI is in slave mode and CRC function is enable, the CRC calculator is sensitive to input SCK clock whether SPI is enable or not. The software must enable CRC only when the clock is stable to avoid wrong CRC calculation. And when SPI works as a slave, the NSS internal signal needs to be kept low between the data phase and CRC phase.

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).

25.6. SPI interrupts

25.6.1. Status flags

- **Transmit buffer empty flag (TBE)**

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI_DATA register.

- **Receive buffer not empty flag (RBNE)**

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI_DATA register.

- **SPI Transmitting On-Going flag (TRANS)**

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn't generate any interrupt.

25.6.2. Error conditions

- **Configuration Fault 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 SWNSS 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 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.

- **Rx Overrun Error (RXORERR)**

The RXORERR bit is set if a data is received when the RBNE is set. That means, the last data has not been read out and the newly incoming data is received. The receive buffer contents won't be covered with the newly incoming data, so the newly incoming data is lost.

- **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.

- **CRC Error (CRCERR)**

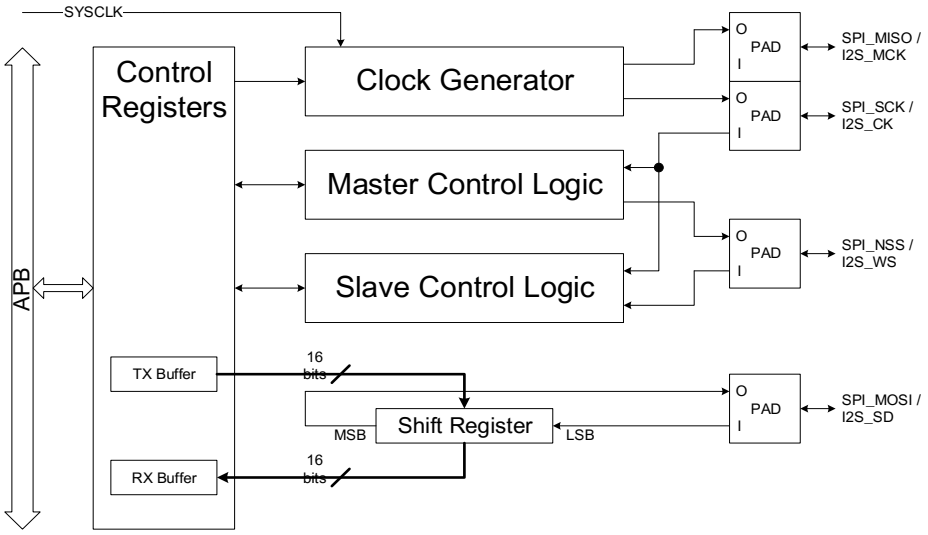
When the CRCEN bit is set, the CRC value received in the SPI_RCRC register will be compared with the CRC value received immediately after the last frame of data. The CRCERR will set when they are different.

Table 25-6. SPI interrupt requests

Flag	Description	Clear Method	interrupt enable bit
TBE	Transmit buffer empty	Write SPI_DATA register.	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
CONFERR	Configuration Fault Error	Read or write SPI_STAT register, then write SPI_CTL0 register.	ERRIE
RXORERR	Rx Overrun Error	Read SPI_DATA register, then read SPI_STAT register.	
CRCERR	CRC error	Write 0 to CRCERR bit	
FERR	TI Mode Format Error	Write 0 to FERR bit	

25.7. I2S block diagram

Figure 25-13. Block diagram of I2S



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 TX buffer and RX buffer. The clock generator is used to produce I2S communication clock in master mode. 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.

25.8. 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. In SPI0, SPI3 and SPI4, MCK shares the same pin with SPI_MISO. In SPI1 and SPI2, MCK has a dedicated pin. MCK is an optional signal for I2S interface. It produces a frequency rate equal to 256 x Fs, and Fs is the audio sampling frequency.

25.9. I2S function overview

25.9.1. 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. The data buffer for transmission and reception is 16-bit wide. In the case that the data length is 24 bits or 32 bits, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In the case that the data length is 16 bits, only one write or read operation to or from the SPI_DATA register is needed to complete a frame. When using 16-bit data packed in 32-bit frame, 16-bit 0 is inserted by hardware automatically to extend the data to 32-bit format.

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 25-14. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

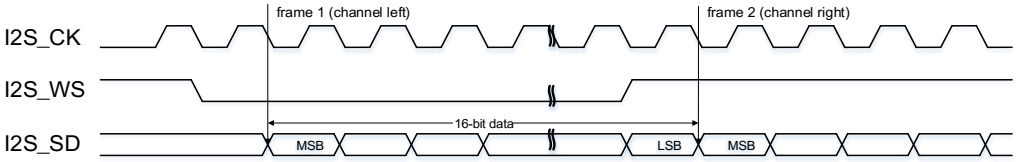
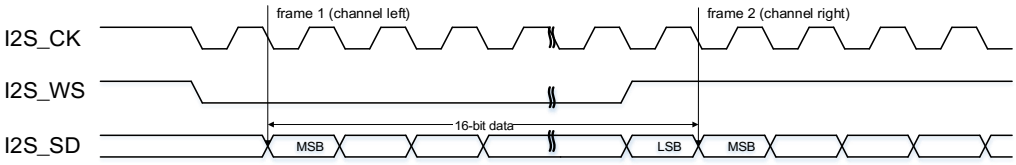


Figure 25-15. I2S Philips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)



When the packet type is 16-bit data packed in 16-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame.

Figure 25-16. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

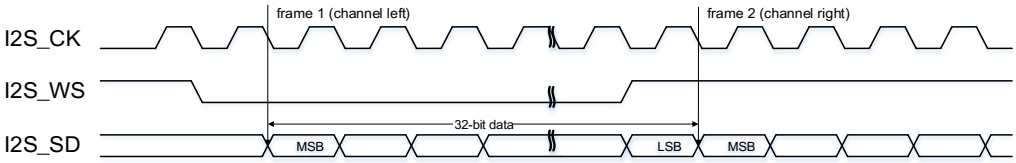
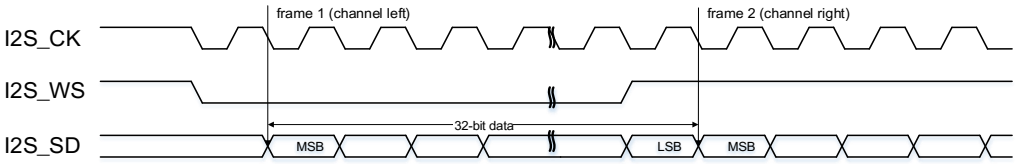


Figure 25-17. I2S Philips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)



When the packet type is 32-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete the transmission of a frame. In transmission mode, if a 32-bit data is going to be sent, the first data written to the SPI_DATA register should be the higher 16 bits, and the second one should be the lower 16 bits. In reception mode, if a 32-bit data is received, the first data read from the SPI_DATA register should be higher 16 bits, and the second one should be the lower 16 bits.

Figure 25-18. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

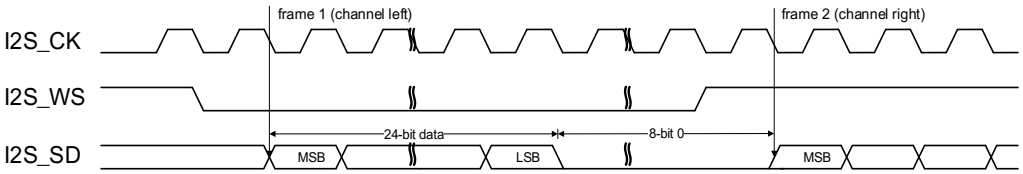
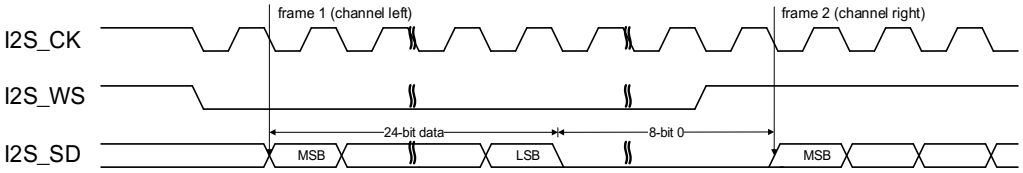


Figure 25-19. I2S Philips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI_DATA register should be the higher 16 bits: D[23:8], and the second one should be a 16-bit data. The higher 8 bits of this 16-bit data should be D[7:0] and the lower 8 bits can be any value. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI_DATA register is D[23:8], and the second one is a 16-bit data. The higher 8 bits of this 16-bit data are D[7:0] and the lower 8 bits are zeros.

Figure 25-20. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

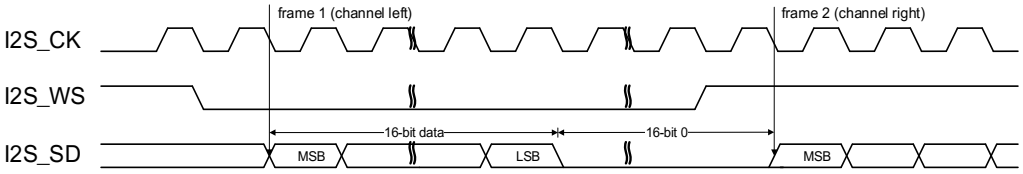
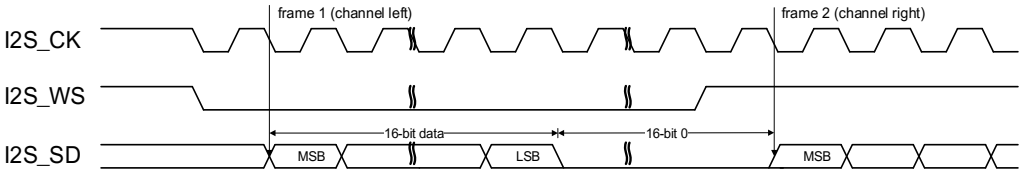


Figure 25-21. I2S Philips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame. The remaining 16 bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

MSB justified standard

For MSB justified standard, I2S_WS and I2S_SD are updated on the falling edge of I2S_CK. The SPI_DATA register is handled in the exactly same way as that for I2S Philips standard. The timing diagrams for each configuration are shown below.

Figure 25-22. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

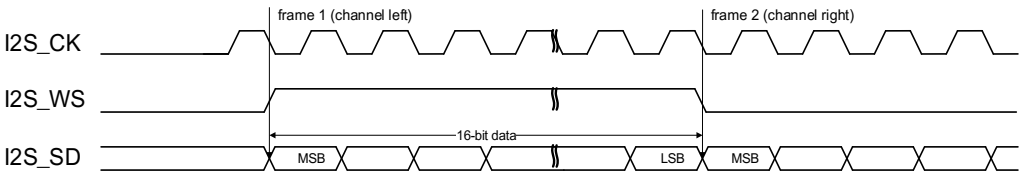


Figure 25-23. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

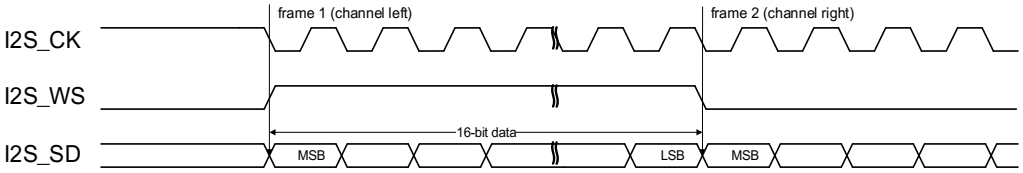


Figure 25-24. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

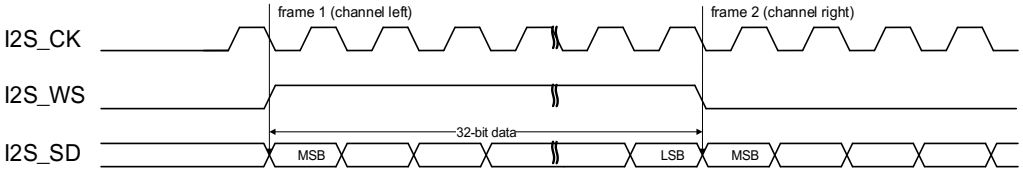


Figure 25-25. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

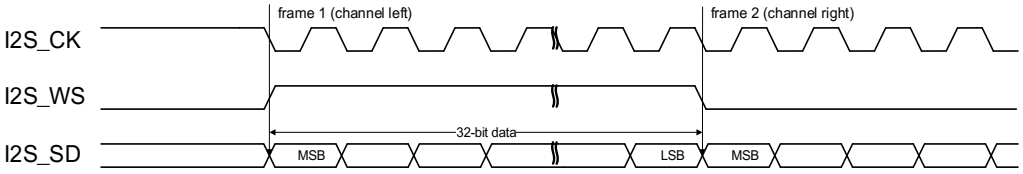


Figure 25-26. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

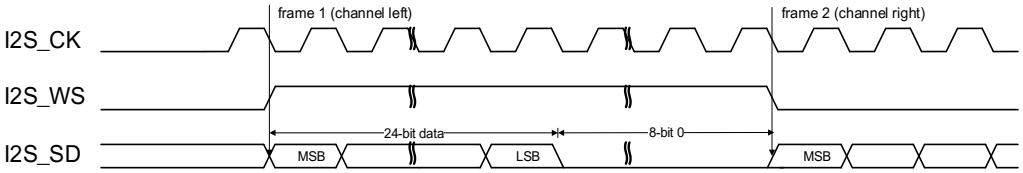


Figure 25-27. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

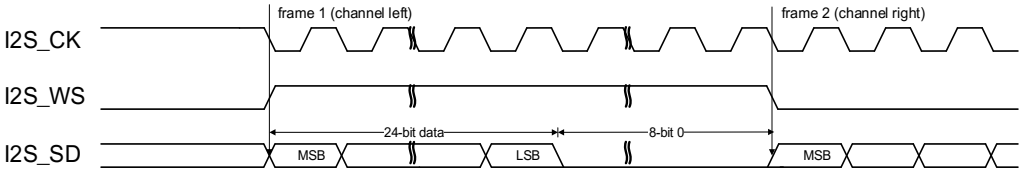


Figure 25-28. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

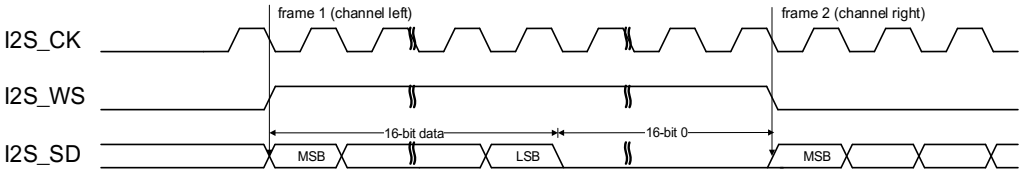
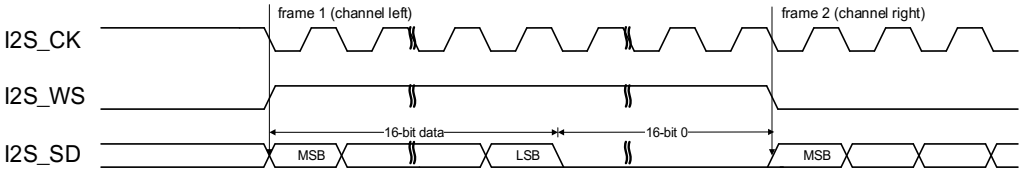


Figure 25-29. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



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 25-30. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

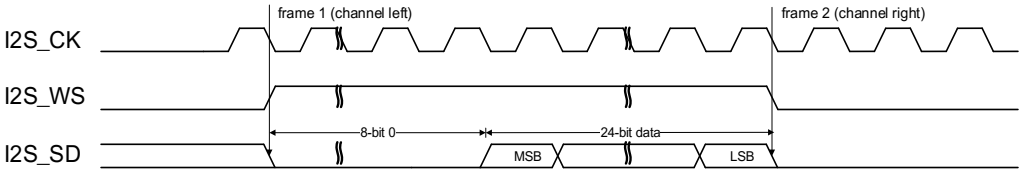
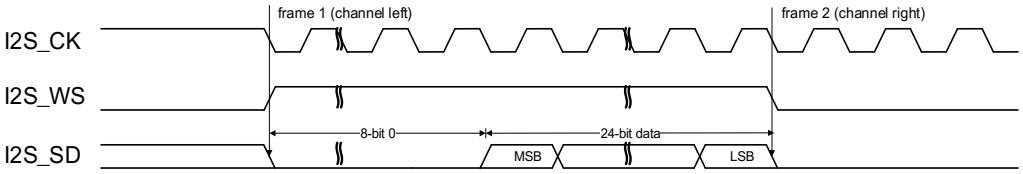


Figure 25-31. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)



When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI_DATA register are needed to complete the transmission of a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI_DATA register should be a 16-bit data. The higher 8 bits of the 16-bit data can be any value and the lower 8 bits should be D[23:16]. The second data written to the SPI_DATA register should be D[15:0]. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI_DATA register is a 16-bit data. The high 8 bits of this 16-bit data are zeros and the lower 8 bits are D[23:16]. The second data read from the SPI_DATA register is D[15:0].

Figure 25-32. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

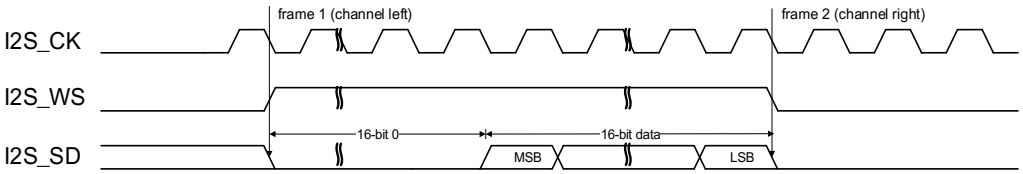
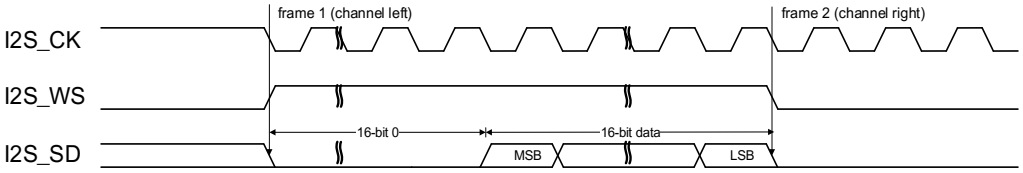


Figure 25-33. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI_DATA register is needed to complete the transmission of a frame. The remaining 16 bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

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_DATA 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 25-34. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)

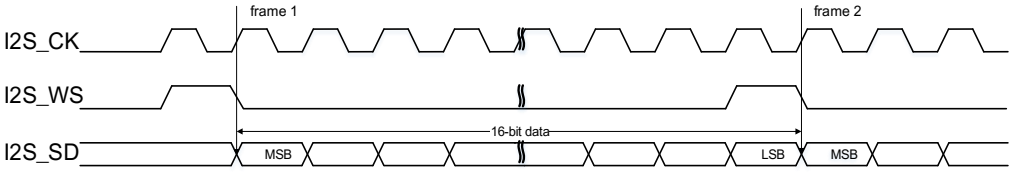


Figure 25-35. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

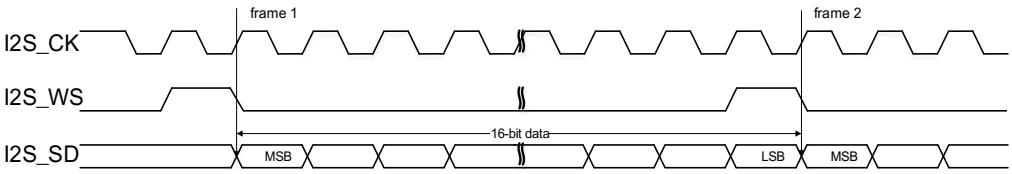


Figure 25-36. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

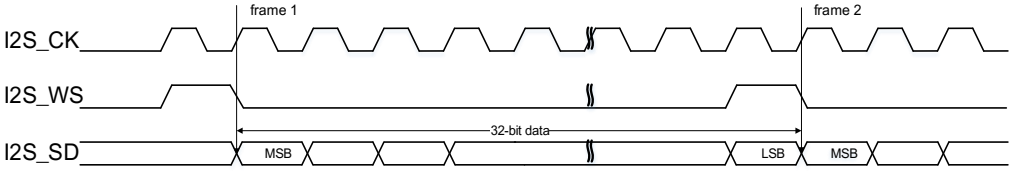


Figure 25-37. PCM standard short frame synchronization mode timing diagram

(DTLEN=10, CHLEN=1, CKPL=1)

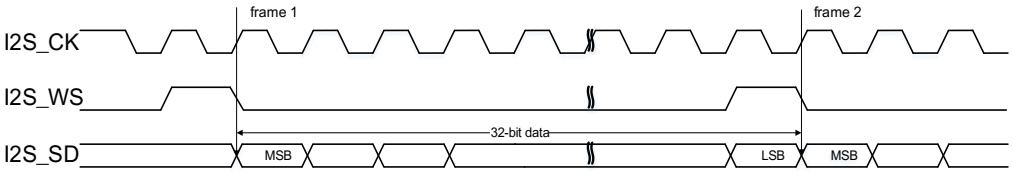


Figure 25-38. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

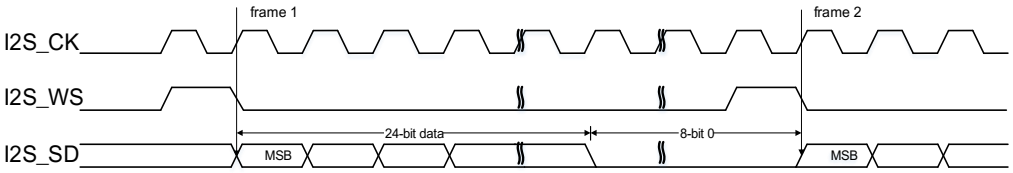


Figure25-39. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

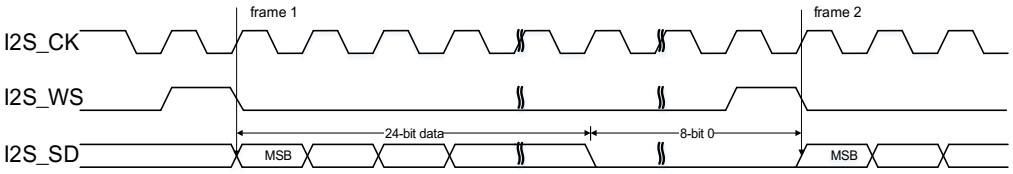


Figure 25-40. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

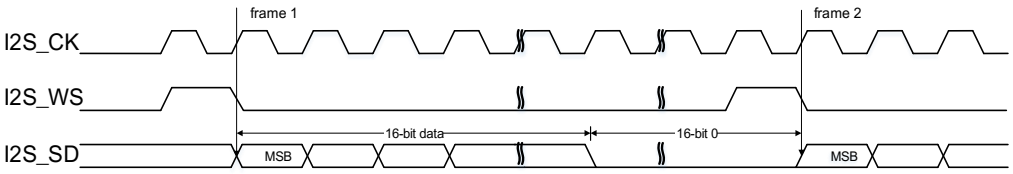
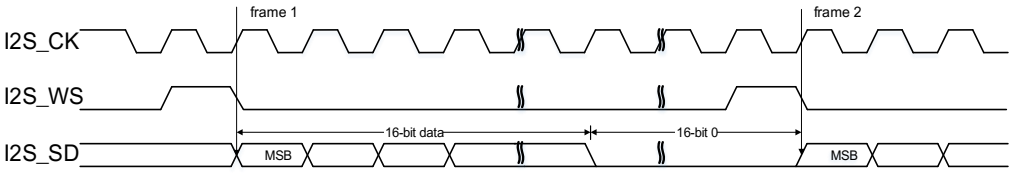


Figure 25-41. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



The timing diagrams for each configuration of the long frame synchronization mode are shown below.

Figure 25-42. PCM standard long frame synchronization mode timing diagram

(DTLEN=00, CHLEN=0, CKPL=0)

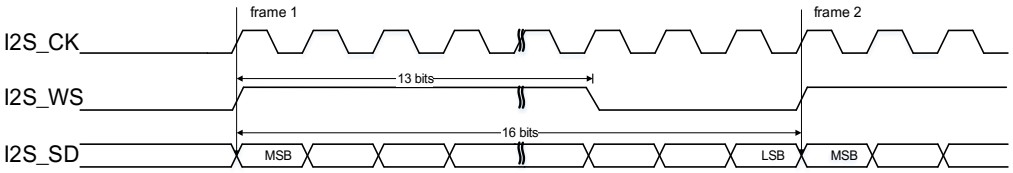


Figure25-43. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)

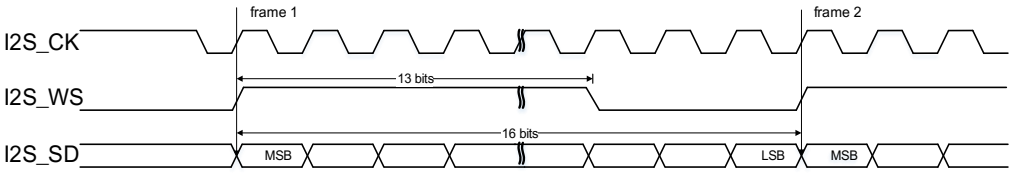


Figure 25-44. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)

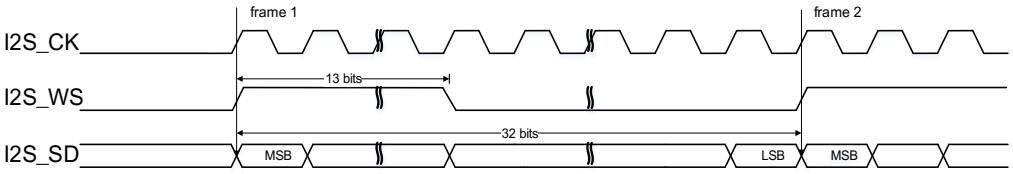


Figure 25-45. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)

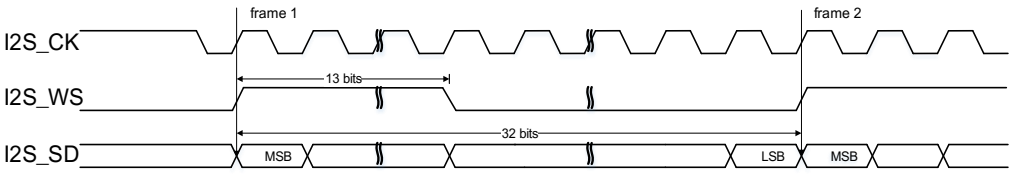


Figure 25-46. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)

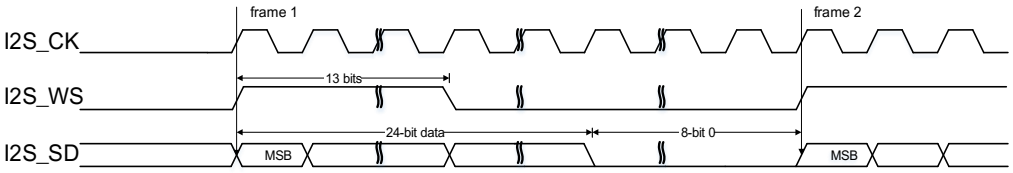


Figure 25-47. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)

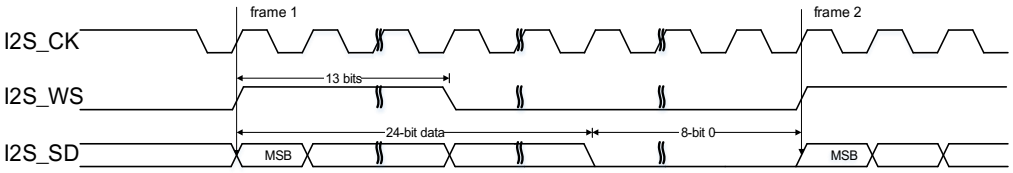


Figure 25-48. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)

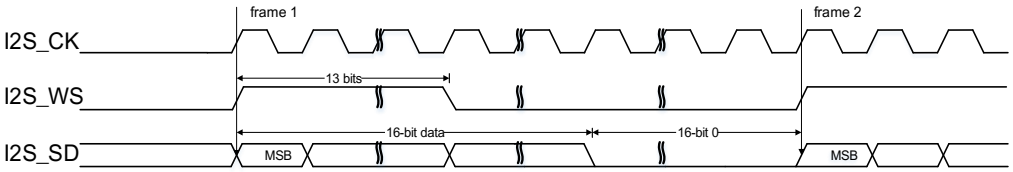
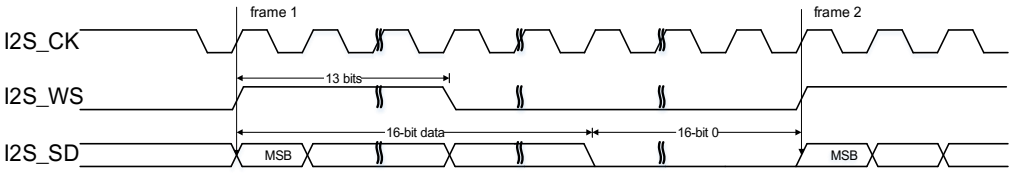
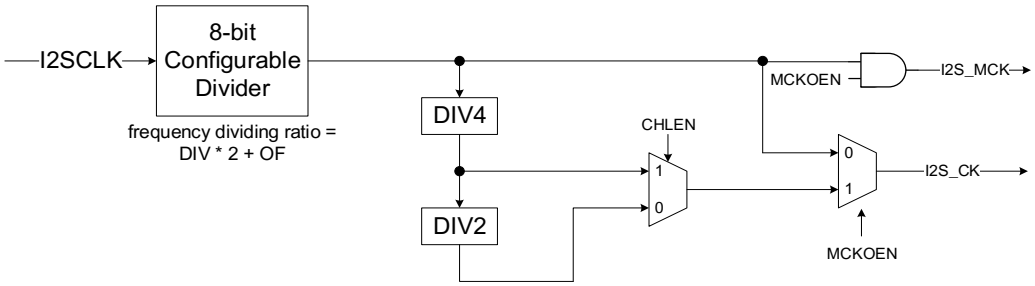


Figure 25-49. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)



25.9.2. I2S clock

Figure 25-50. Block diagram of I2S clock generator



The block diagram of I2S clock generator is shown as ***Figure 25-50. Block diagram of I2S clock generator***.The I2S interface clocks are configured by the DIV bits, the OF bit, the MCKOEN bit in the SPI_I2SPSC register and the CHLEN bit in the SPI_I2SCTL register. The source clock is the system clock(CK_SYS). The I2S bitrate can be calculated by the formulas shown in ***Table 25-7. I2S bitrate calculation formulas***.

Table 25-7. I2S bitrate calculation formulas

MCKOEN	CHLEN	Formula
0	0	I2SCLK / (DIV * 2 + OF)
0	1	I2SCLK / (DIV * 2 + OF)
1	0	I2SCLK / (8 * (DIV * 2 + OF))
1	1	I2SCLK / (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 25-8. Audio sampling frequency calculation formulas*.

Table 25-8. Audio sampling frequency calculation formulas

MCKOEN	CHLEN	Formula
0	0	I2SCLK / (32 * (DIV * 2 + OF))
0	1	I2SCLK / (64 * (DIV * 2 + OF))
1	0	I2SCLK / (256 * (DIV * 2 + OF))
1	1	I2SCLK / (256 * (DIV * 2 + OF))

The source of I2S clock can be either from PLLI2S or an external I2S_CKIN pin, and this is programmable in RCU. Software should carefully calculate the factor of I2S and PLLI2S to get the most accurate audio sampling frequency. If PLLI2S cannot meet the application's precision demand, an external precise I2S clock can be imported from I2S_CKIN pin.

Note: The configuration value of the I2S serial clock must be set to less than 1/6 of the PCLK clock (excluding 1/6).

25.9.3. 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 *Table 25-9. Direction of I2S interface signals for each operation mode*.

Table 25-9. Direction of I2S interface signals for each operation mode

Operation mode	I2S_MCK	I2S_CK	I2S_WS	I2S_SD	I2S_ADD_SD(2)
Master transmission	output or NU(1)	output	output	output	NU(1)
Master reception	output or NU(1)	output	output	input	NU(1)
Slave transmission	input or NU(1)	input	input	output	NU(1)
Slave reception	input or NU(1)	input	input	input	NU(1)
Full-duplex	output or NU(1)	output	output	output or input	Input or output

1. NU means the pin is not used by I2S and can be used by other functions.
2. To support full-duplex operation in I2S1 and I2S2, there are two extra I2S modules in chip called I2S_ADD1 and I2S_ADD2. I2S_ADD_SD is the data pin of I2S_ADD module. Full-duplex will be described later in this chapter.

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 working in slave mode, only step 2, step 3, step 4 and step 5 should be done.

- Step 1: Configure the DIV[7:0] bits, the OF bit, and the MCKOEN bit in the SPI_I2SPSC register, in order to define the I2S bitrate and 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 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, in order to define the I2S feature.
- Step 4: Configure the TBEIE bit, the RBNEIE bit, the ERRIE bit, the DMATEN bit, and the DMAREN bit in the SPI_CTL1 register, in order to select the potential interrupt sources and the DMA capabilities. This step is optional.
- Step 5: Set the I2SEN bit in the SPI_I2SCTL register to enable I2S.

I2S master transmission sequence

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates that the transmit buffer is empty, and an interrupt will be generated if the TBEIE bit in the SPI_CTL1 register is set. At the beginning, the transmit buffer is empty (TBE is high) and no transmission sequence is processing in the shift register. When a half word is written to the SPI_DATA register (TBE goes low), the data is transferred from the transmit buffer to the shift register (TBE goes high) immediately. At the moment, the transmission sequence begins.

The data is parallel loaded into the 16-bit shift register, and shifted out serially to the I2S_SD pin, MSB first. The next data should be written to the SPI_DATA register, when the TBE flag is high. After a write operation to the SPI_DATA register, the TBE flag goes low. When the current transmission finishes, the data in the transmit buffer is loaded into the shift register, and the TBE flag goes back high. Software should write the next audio data into SPI_DATA register before the current data finishes, otherwise, the audio data transmission is not continuous.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the data to transfer belongs. The I2SCH flag is refreshed at the moment when the TBE flag goes high. At the beginning, the I2SCH flag is low, indicating the left channel data should be written to the SPI_DATA register.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

I2S master reception sequence

The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE

flag indicates the receive buffer is not empty, and an interrupt will be generated if the RBNEIE bit in the SPI_CTL1 register is set. The reception sequence begins immediately when the I2SEN bit in the SPI_I2SCTL register is set. At the beginning, the receive buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the receive buffer (RBNE goes high). The data should be read from the SPI_DATA register, when the RBNE flag is high. After a read operation to the SPI_DATA register, the RBNE flag goes low. It is mandatory to read the SPI_DATA register before the end of the next reception. Otherwise, reception overrun error occurs. When overrun error occurred, the RXORERR flag is set. And an interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In this case, it is necessary to switch off and then switch on I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the received data belongs. The I2SCH flag is refreshed at the moment when the RBNE flag goes high.

Different sequences are used to disable the I2S in different standards, data length and channel length. The sequences for each case are described below.

- 16-bit data packed in 32-bit frame in the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD = 10)

1. Wait for the second last RBNE
2. Then wait 17 I2S CK clock (clock on I2S_CK pin) cycles
3. Clear the I2SEN bit

- 16-bit data packed in 32-bit frame in the audio standards except the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD is not equal to 10)

1. Wait for the last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

- For all other cases

1. Wait for the second last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

I2S slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The transmission sequence begins when the external master sends the clock and when the I2S_WS signal requests the transfer of data. The data has to be written to the SPI_DATA register before the master initiates the communication. Software should write the next audio data into SPI_DATA register before the current data finishe. Otherwise, transmission underrun error occurs. The TXURERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI_CTL1 register is set. In this case, it is mandatory to

switch off and switch on I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

I2S slave reception sequence

The reception sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The reception sequence begins when the external master sends the clock and when the I2S_WS signal indicates a start of the data transfer. In slave mode, I2SCH is sensitive to the I2S_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

I2S Full-Duplex Mode

A single I2S only supports one-way transmission: transmit or receive mode. I2S full-duplex is supported by using an extra I2S module: I2S_ADD simultaneously with I2S. I2S_ADD module has the same function with I2S module, but can only work in slave mode. There are two I2S_ADD modules: I2S_ADD1 and I2S_ADD2, so only I2S1 and I2S2 support full-duplex mode. I2S_ADD's I2S_CK and I2S_WS are internally connected to its respective I2S's respective ports. I2S_ADD's I2S_SD pin is mapped to respective I2S's SPI_MISO pin.

In order to work in full-duplex mode, application should enable the I2S module as well as its corresponding I2S_ADD module. I2S supports two full-duplex modes: master mode and salve mode.

In master full-duplex mode, software should set I2S as a master, and I2S_ADD as a slave. Then I2S_ADD's WS and SCK signals come from the master I2S.

In slave full-duplex mode, software should set both I2S and I2S_ADD as slaves. Then, the WS and CK signals of both I2S_ADD and I2S come from external.

Application may configure I2S into either a transmitter or a receiver and thus, configure I2S_ADD into opposite data direction. During transmission, software should operate registers and handle interrupts for both I2S and I2S_ADD to make a full-duplex transmission.

25.9.4. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC feature is not available in I2S mode.

25.10. I2S interrupts

25.10.1. Status flags

There are four status flags implemented in the SPI_STAT register, including TBE, RBNE, TRANS and I2SCH. The user can use them to fully monitor the state of the I2S bus.

- **Transmit buffer empty flag (TBE)**

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI_DATA register.

- **Receive buffer not empty flag (RBNE)**

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI_DATA register.

- **I2S Transmitting On-Going flag (TRANS)**

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn't generate any interrupt.

- **I2S channel side flag (I2SCH)**

This flag indicates the channel side information of the current transfer and has no meaning in PCM mode. It is updated when TBE rises in transmission mode or RBNE rises in reception mode. This flag doesn't generate any interrupt.

25.10.2. Error conditions

There are three error conditions:

- **Transmission Underrun Error Flag (TXURERR)**

In the slave transmit mode, when the valid SCK signal starts transmitting, if the transmit buffer is empty, TXURERR will be set.

- **Reception Overrun Error Flag (RXORERR)**

This condition occurs when the receive buffer is full and a newly incoming data has been completely received. When overrun occurs, the data in receive buffer is not updated and the newly incoming data is lost.

- **Format Error (FERR)**

In slave I2S mode, the I2S monitors the I2S_WS signal and an error flag will be set if I2S_WS toggles at an unexpected position.

I2S interrupt events and corresponding enabled bits are summed up in *Table 25-10. I2S interrupt*.

Table 25-10. I2S interrupt

Flag Name	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
TXURERR	Transmission underrun error	Read SPI_STAT register	ERRIE
RXORERR	Reception overrun error	Read SPI_DATA register and then read SPI_STAT register.	
FERR	I2S Format Error	Read SPI_STAT register	

25.11. Register definition

I2S1_add base address: 0x4000 3400

SPI1/I2S1 base address: 0x4000 3800

SPI2/I2S2 base address: 0x4000 3C00

I2S2_add base address: 0x4000 4000

SPI0 base address: 0x4001 3000

SPI3 base address: 0x4001 3400

SPI4 base address: 0x4001 5000

SPI5 base address: 0x4001 5400

25.11.1. Control register 0 (SPI_CTL0)

Address offset: 0x00
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).

This register has no meaning in I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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
BDEN	BDOEN	CRCEN	CRCNT	FF16	RO	SWNSS EN	SWNSS	LF	SPIEN	PSC [2:0]			MSTMOD	CKPL	CKPH
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	BDEN	Bidirectional enable 0: 2 line unidirectional transmit mode 1: 1 line bidirectional transmit mode. The information transfers between the MOSI pin in master and the MISO pin in slave.
14	BDOEN	Bidirectional transmit output enable When BDEN is set, this bit determines the direction of transfer. 0: Work in receive-only mode 1: Work in transmit-only mode
13	CRCEN	CRC calculation enable 0: CRC calculation is disabled

		1: CRC calculation is enabled.
12	CRCNT	CRC next transfer 0: Next transfer is Data 1: Next transfer is CRC value (TCR) When the transfer is managed by DMA, CRC value is transferred by hardware. This bit should be cleared. In full-duplex or transmit-only mode, set this bit after the last data is written to SPI_DATA register. In receive only mode, set this bit after the second last data is received.
11	FF16	Data frame format 0: 8-bit data frame format 1: 16-bit data frame format
10	RO	Receive only When BDEN is cleared, this bit determines the direction of transfer. 0: Full-duplex 1: Receive-only
9	SWNSSEN	NSS software mode selection 0: NSS hardware mode. The NSS level depends on NSS pin. 1: NSS software mode. The NSS level depends on SWNSS bit. This bit has no meaning in SPI TI mode.
8	SWNSS	NSS pin selection in NSS software mode 0: NSS pin is pulled low 1: NSS pin is pulled high This bit has an effect only when the SWNSSEN bit is set. This bit has no meaning in SPI TI mode.
7	LF	LSB first mode 0: Transmit MSB first 1: Transmit LSB first This bit has no meaning in SPI TI mode.
6	SPIEN	SPI enable 0: SPI peripheral is disabled 1: SPI peripheral is enabled
5:3	PSC[2:0]	Master clock prescaler selection 000: PCLK/2 100: PCLK/32 001: PCLK/4 101: PCLK/64 010: PCLK/8 110: PCLK/128 011: PCLK/16 111: PCLK/256 PCLK means PCLK2 when using SPI0, SPI3, SPI4 and SPI5 or PCLK1 when using SPI1 and SPI2.

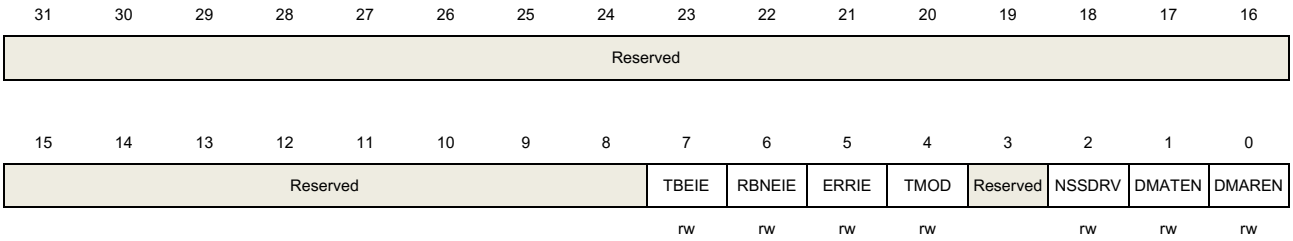
2	MSTMOD	Master mode enable 0: Slave mode 1: Master mode
1	CKPL	Clock polarity selection 0: CLK pin is pulled low when SPI is idle 1: CLK pin is pulled high when SPI is idle
0	CKPH	Clock phase selection 0: Capture the first data at the first clock transition. 1: Capture the first data at the second clock transition

25.11.2. Control register 1 (SPI_CTL1)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	TBEIE	Transmit buffer empty interrupt enable 0: TBE interrupt is disabled. 1: TBE interrupt is enabled. An interrupt is generated when the TBE bit is set.
6	RBNEIE	Receive buffer not empty interrupt enable 0: RBNE interrupt is disabled. 1: RBNE interrupt is enabled. An interrupt is generated when the RBNE bit is set.
5	ERRIE	Errors interrupt enable. 0: Error interrupt is disabled. 1: Error interrupt is enabled. An interrupt is generated when the CRCERR bit or the CONFERR bit or the RXORERR bit or the TXURERR bit is set.
4	TMOD	SPI TI mode enable 0: SPI TI Mode Disabled. 1: SPI TI Mode Enabled.
3	Reserved	Must be kept at reset value.
2	NSSDRV	Drive NSS output

		0: NSS output is disabled. 1: NSS output is enabled. If the NSS pin is configured as output, the NSS pin is pulled low in master mode when SPI is enabled. If the NSS pin is configured as input, the NSS pin should be pulled high in master mode, and this bit has no effect.
1	DMATEN	Transmit buffer DMA enable 0: Transmit buffer DMA is disabled 1: Transmit buffer DMA is enabled. When the TBE bit in SPI_STAT is set, it will generate a DMA request at corresponding DMA channel.
0	DMAREN	Receive buffer DMA enable 0: Receive buffer DMA is disabled 1: Receive buffer DMA is enabled, when the RBNE bit in SPI_STAT is set, it will generate a DMA request at corresponding DMA channel.

25.11.3. Status register (SPI_STAT)

Address offset: 0x08

Reset value: 0x0000 0002

This register can be accessed by byte (8-bit) or 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							FERR	TRANS	RXORERR	CONFERR	CRCERR	TXURERR	I2SCH	TBE	RBNE
							rc_w0	r	r	r	rc_w0	r	r	r	r

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	FERR	Format error SPI TI Mode: 0: No TI Mode format error 1: TI Mode format error occurs. I2S Mode: 0: No I2S format error 1: I2S format error occurs. This bit is set by hardware and is able to be cleared by writing 0.
7	TRANS	Transmitting on-going bit 0: SPI or I2S is idle. 1: SPI or I2S is currently transmitting and/or receiving a frame This bit is set and cleared by hardware.

6	RXORERR	Reception overrun error bit 0: No reception overrun error occurs. 1: Reception overrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_DATA register followed by a read access to the SPI_STAT register.
5	CONFERR	SPI configuration error 0: No configuration fault occurs 1: Configuration fault occurred (In master mode, the NSS pin is pulled low in NSS hardware mode or SWNSS bit is low in NSS software mode.). This bit is set by hardware and cleared by a read or write operation on the SPI_STAT register followed by a write access to the SPI_CTL0 register. This bit is not used in I2S mode.
4	CRCERR	SPI CRC error bit 0: The SPI_RCRC value is equal to the received CRC data at last. 1: The SPI_RCRC value is not equal to the received CRC data at last. This bit is set by hardware and is able to be cleared by writing 0. This bit is not used in I2S mode.
3	TXURERR	Transmission underrun error bit 0: No transmission underrun error occurs. 1: Transmission underrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_STAT register. This bit is not used in SPI mode.
2	I2SCH	I2S channel side 0: The next data needs to be transmitted or the data just received is channel left. 1: The next data needs to be transmitted or the data just received is channel right. This bit is set and cleared by hardware. This bit is not used in SPI mode, and has no meaning in the I2S PCM mode.
1	TBE	Transmit buffer empty 0: Transmit buffer is not empty 1: Transmit buffer is empty
0	RBNE	Receive buffer not empty 0: Receive buffer is empty 1: Receive buffer is not empty

25.11.4. Data register (SPI_DATA)

Address offset: 0x0C
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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
SPI_DATA[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	SPI_DATA[15:0]	Data transfer register The hardware has two buffers, including transmit buffer and receive buffer. Write data to SPI_DATA will save the data to transmit buffer and read data from SPI_DATA will get the data from receive buffer. When the data frame format is set to 8-bit data, the SPI_DATA[15:8] is forced to 0 and the SPI_DATA[7:0] is used for transmission and reception, transmit buffer and receive buffer are 8-bits. If the Data frame format is set to 16-bit data, the SPI_DATA[15:0] is used for transmission and reception, transmit buffer and receive buffer are 16-bit.

25.11.5. CRC polynomial register (SPI_CRCPOLY)

Address offset: 0x10
Reset value: 0x0000 0007

This register can be accessed by byte (8-bit) or 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
CRCPOLY[15:0]															
rw															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CRCPOLY[15:0]	CRC polynomial register This register contains the CRC polynomial and it is used for CRC calculation. The default value is 0007h.

25.11.6. RX CRC register (SPI_RCRC)

Address offset: 0x14
Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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
RCRC[15:0]															
r															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RCRC[15:0]	RX CRC register When the CRCERRN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and saves them in RCRC register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in RCRC[7:0]. When the Data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in RCRC[15:0]. The hardware computes the CRC value after each received bit, when the TRANS is set, a read to this register could return an intermediate value. This register is reset when the CRCEN bit or the SPIEN bit in SPI_CTL0 register is cleared.

25.11.7. TX CRC register (SPI_TCRC)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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
TCRC[15:0]															
r															

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	TCRC[15:0]	TX CRC register When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the transmitted bytes and saves them in TCR register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in TCRC[7:0].When the Data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in TCRC[15:0]. The hardware computes the CRC value after each transmitted bit, when the TRANS is set, a read to this register could return an intermediate value. The different frame

format (LF bit of the SPI_CTL0) will get different CRC value.

This register is reset when the CRCEN bit or the SPIEN bit in SPI_CTL0 register is cleared.

25.11.8. I2S control register (SPI_I2SCTL)

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or 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				I2SSEL	I2SEN	I2SOPMOD[1:0]		PCMSMOD	Reserved	I2SSTD[1:0]		CKPL	DTLEN[1:0]		CHLEN
				rw	rw	rw		rw		rw		rw	rw		rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	I2SSEL	I2S mode selection 0: SPI mode 1: I2S mode This bit should be configured when SPI mode or I2S mode is disabled.
10	I2SEN	I2S enable 0: I2S is disabled 1: I2S is enabled This bit is not used in SPI mode.
9:8	I2SOPMOD[1:0]	I2S operation mode 00: Slave transmission mode 01: Slave reception mode 10: Master transmission mode 11: Master reception mode This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
7	PCMSMOD	PCM frame synchronization mode 0: Short frame synchronization 1: long frame synchronization This bit has a meaning only when PCM standard is used. This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
6	Reserved	Must be kept at reset value.

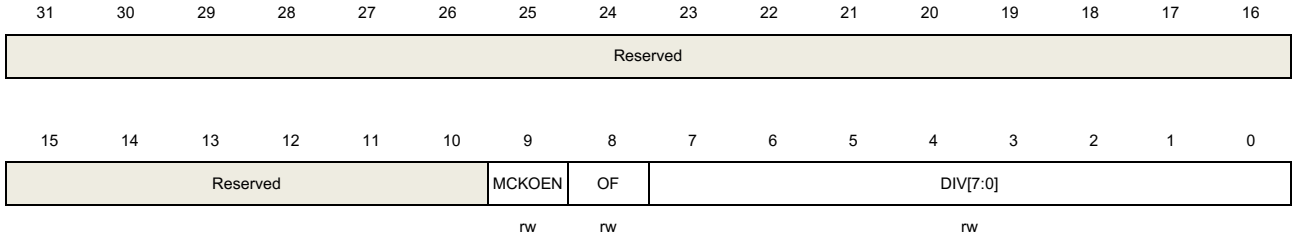
5:4	I2SSTD[1:0]	I2S standard selection 00: I2S Philips standard 01: MSB justified standard 10: LSB justified standard 11: PCM standard These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.
3	CKPL	Idle state clock polarity 0: The idle state of I2S_CK is low level 1: The idle state of I2S_CK is high level This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
2:1	DTLEN[1:0]	Data length 00: 16 bits 01: 24 bits 10: 32 bits 11: Reserved These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.
0	CHLEN	Channel length 0: 16 bits 1: 32 bits The channel length must be equal to or greater than the data length. This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.

25.11.9. I2S clock prescaler register (SPI_I2SPSC)

Address offset: 0x20

Reset value: 0x0000 0002

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	MCKOEN	I2S_MCK output enable

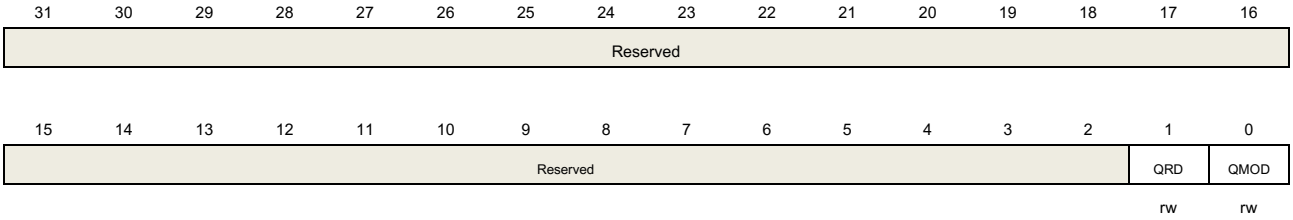
		0: I2S_MCK output is disabled 1: I2S_MCK output is enabled This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
8	OF	Odd factor for the prescaler 0: Real divider value is DIV * 2 1: Real divider value is DIV * 2 + 1 This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
7:0	DIV[7:0]	Dividing factor for the prescaler Real divider value is DIV * 2 + OF. DIV must not be 0. These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.

25.11.10. Quad-SPI mode control register (SPI_QCTL) of SPI5

Address offset: 0x80

Reset value: 0x0000 0000

This register can be accessed by byte (8-bit) or half-word (16-bit) or word (32-bit).



Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	QRD	Quad-SPI mode read select. 0: SPI is in quad wire write mode 1: SPI is in quad wire read mode This bit should be only be configured when SPI is not busy (TRANS bit cleared). This bit is only available in SPI5.
0	QMOD	Quad-SPI mode enable. 0: SPI is in single wire mode 1: SPI is in Quad-SPI mode This bit should only be configured when SPI is not busy (TRANS bit cleared). This bit is only available in SPI5.

26. Serial Audio Interface (SAI)

26.1. 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.

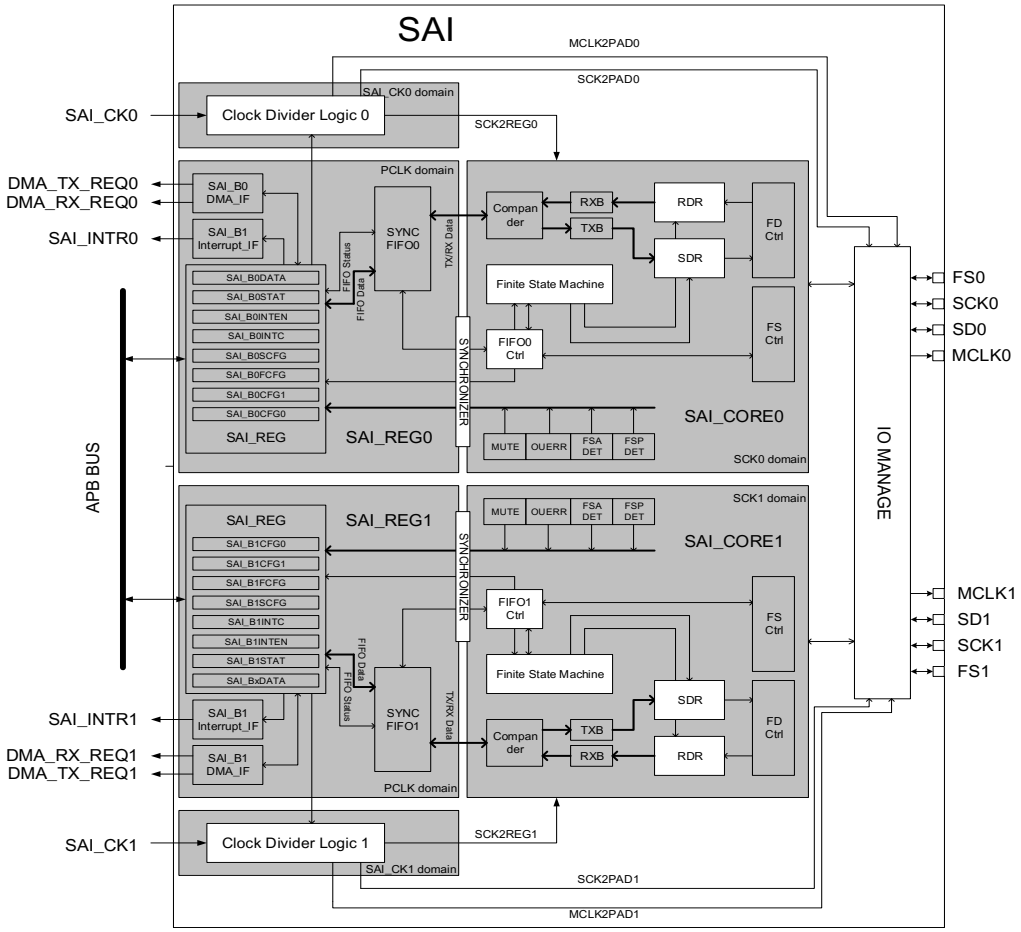
26.2. Characteristics

- 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.
- 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

26.3. Function overview

26.3.1. Block diagram

Figure 26-1. block diagram



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

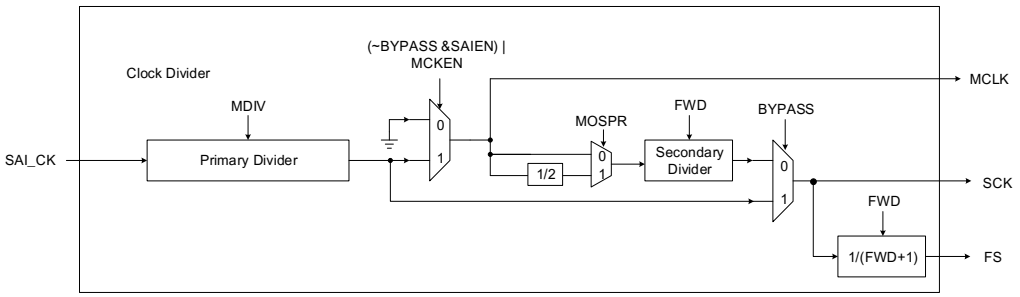
in AC'97 protocol, FS is forced to be an output, independent of master/slave configurations. The Serial Data (SD) IO pin is configured as output in transition and input in reception.

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.

26.3.2. 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 (refer to ***Reset and clock unit (RCU)*** chapter) 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 26-2. 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.

$$f_{MCLK} = \begin{cases} \frac{f_{SAI_CK}}{MDIV}, & MDIV \mathrel{!}= 0 \\ f_{SAI_CK}, & MDIV = 0 \end{cases} \quad (26\text{-}1)$$

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.

$$f_{SCK} = \frac{f_{SAI_CK} \times (FWD + 1)}{MDIV \times (MOSPR + 1) \times 256} \quad (26\text{-}2)$$

The frame synchronization frequency is given by:

$$f_{FS} = \frac{f_{SAI_CK}}{MDIV \times (MOSPR + 1) \times 256} \quad (26\text{-}3)$$

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 26-1. Commonly used audio sampling rate*.

Table 26-1. Commonly used audio sampling rate

SAI_CK Clock Rate	Standard Audio Sampling Rate	Master Clock Divide Ratio
192kHz x 256	192 kHz	MDIV = 0
	96 kHz	MDIV = 1
	48 kHz	MDIV = 2
	16 kHz	MDIV = 6
	8 kHz	MDIV = 12
44.1kHz x 256	44.1 kHz	MDIV = 0
	22.05 kHz	MDIV = 1
	11.025 kHz	MDIV = 2
f_{MCLK}	f_{MCLK}	MDIV = 0

26.3.3. 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.

26.3.4. SAI synchronization mode

Internal synchronization is supported at audio sub-block level.

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.

There is only one SAI in this series, and not support external synchronization mode(SYNO and SYNI in SAI_SYNCFG register should be keep at reset value, SYNCMOD in SAI_BxCFG0 can not be set to 2'b10).

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.

The slave sub-module cannot serve as a synchronization signal source and can only receive FS and SCK signals from the master module.

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..

26.3.5. 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.

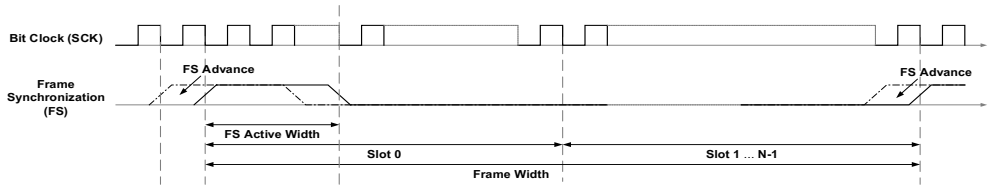
Frame synchronization advancement

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 26-3 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.

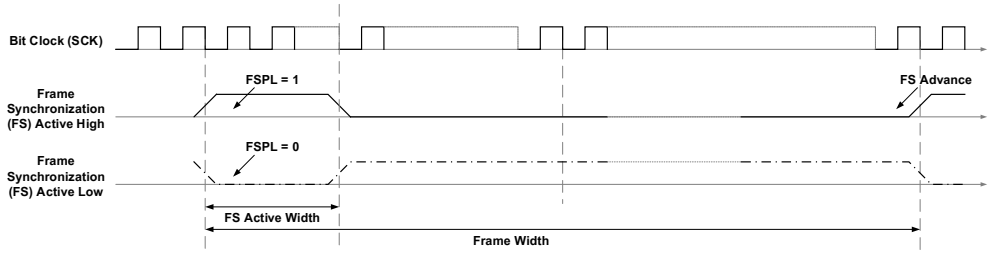
Figure 26-3 FS active width



Frame synchronization polarity

Frame synchronization active level could be configured through FSPL control filed in SAI_BxFCFG register, as shown in the ***Figure 26-4 FS polarity***.

Figure 26-4 FS polarity

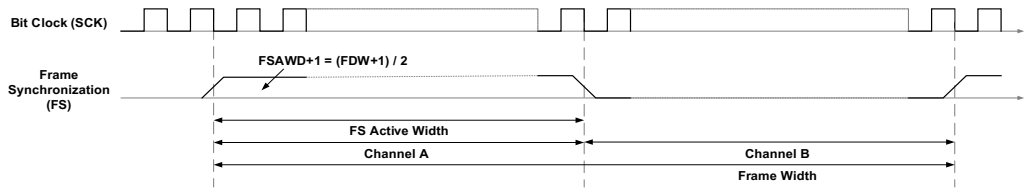


Frame synchronization function

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 26-5 FS function***, otherwise the audio sub-block behavior is not guaranteed. When FSFUNC is set to 0, FS only represents frame start.

Figure 26-5 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 cleared, 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.

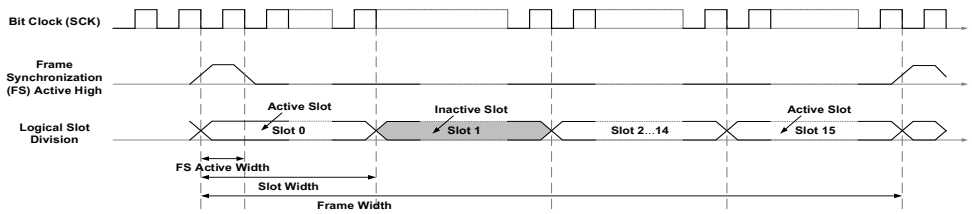
26.3.6. Slot configuration

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 26-6 Slot activation***.

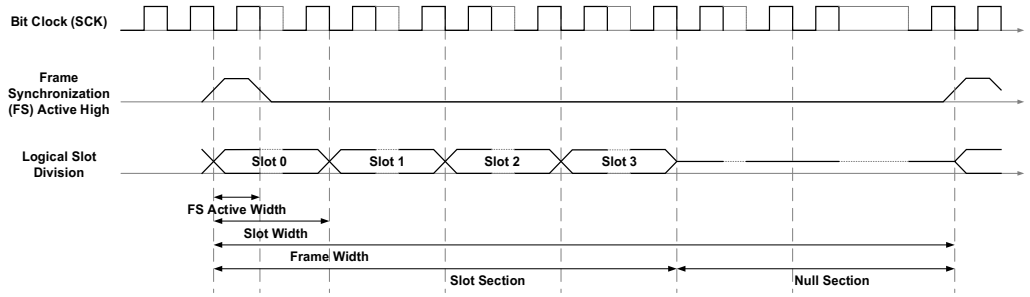
Figure 26-6 Slot activation



Slot distribution

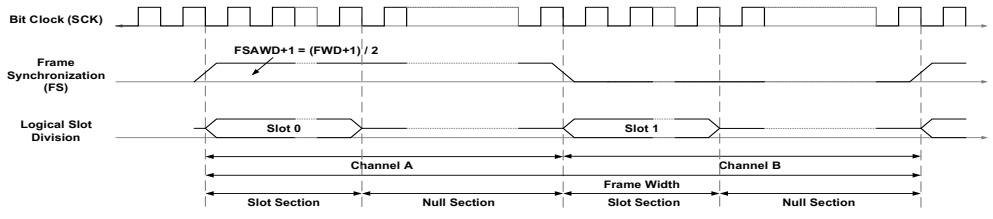
There can be not slot distribution under a specific circumstance that the product of slot number and slot width is less than frame width. Slot section is defined where there is slot distribution; otherwise, it is called a null section. When FSFUNC set to 0, where FS only signals frame start, the null section follows the last slot until the start of the next frame, as shown in the ***Figure 26-7 Slot distribution when FUNC = 0.***

Figure 26-7 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 26-8 Slot distribution when FUNC = 1

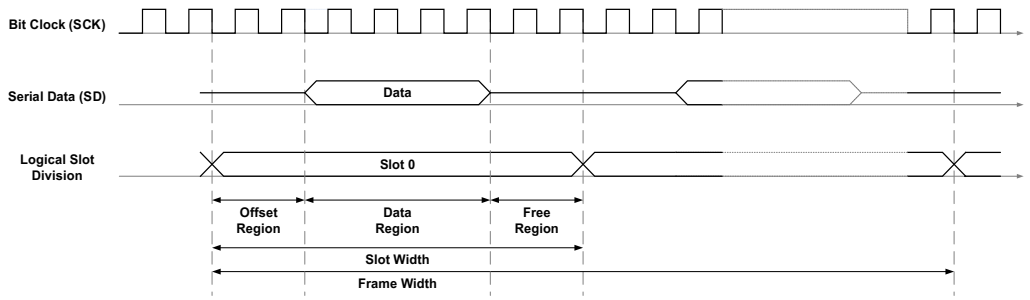


Serial data output management on inactive slots

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,

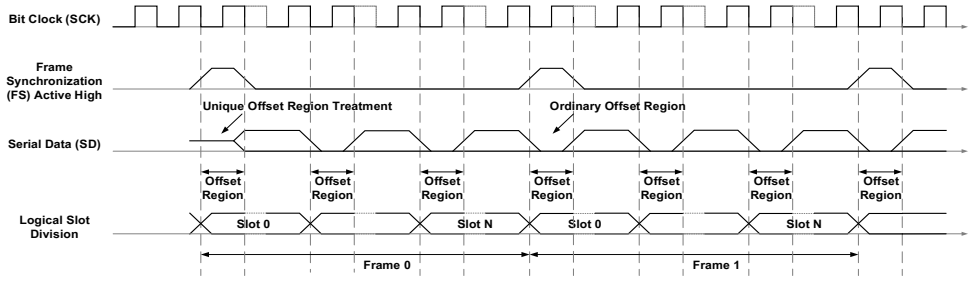
namely offset, data and free region used in this manual is depicted in the *Figure 26-9 Slot partition convention*.

Figure 26-9 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 26-10 Offset region treatment* shows the differences.

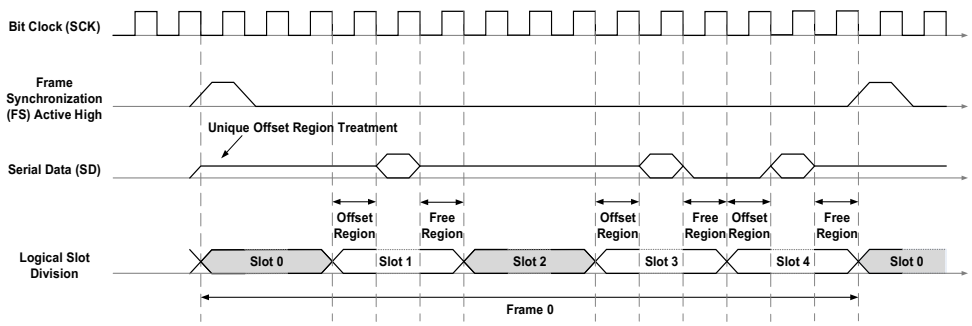
Figure 26-10 Offset region treatment



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 26-11 SD output management*.

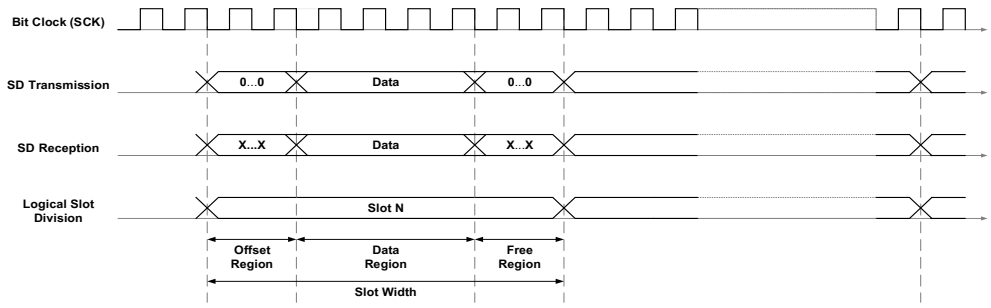
Figure 26-11 SD output management



26.3.7. 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 ***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 26-12 Data configuration***.

Figure 26-12 Data configuration



26.3.8. 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 26-2 FIFO request generation conditions*** and cleared if the underlying conditions are not satisfied.

Table 26-2 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.

26.3.9. 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.

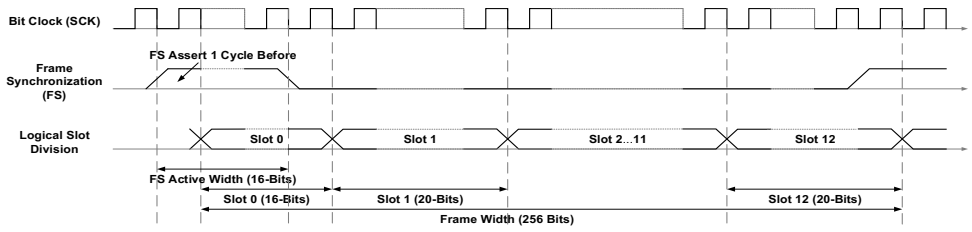
Bit2 of the TAG (slot 0) is reserved, no matter what value is written into the TAG, bit2 of the send data is 0.

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.

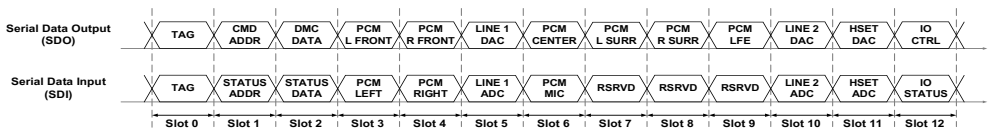
Frame synchronization active edge is asserted 1 clock cycle before the first bit of data, as shown in the ***Figure 26-13 AC'97 slot partition***.

Figure 26-13 AC'97 slot partition



The ***Figure 26-14 AC'97 tag definition*** give an overview of AC'97 slot partition.

Figure 26-14 AC'97 tag definition



The *Table 26-3 AC'97 transmitter tag definition* summarizes each slot's definition and meaning.

When AC'97 link controller acting as transmitter.

Table 26-3 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 26-4 AC'97 receiver tag definition

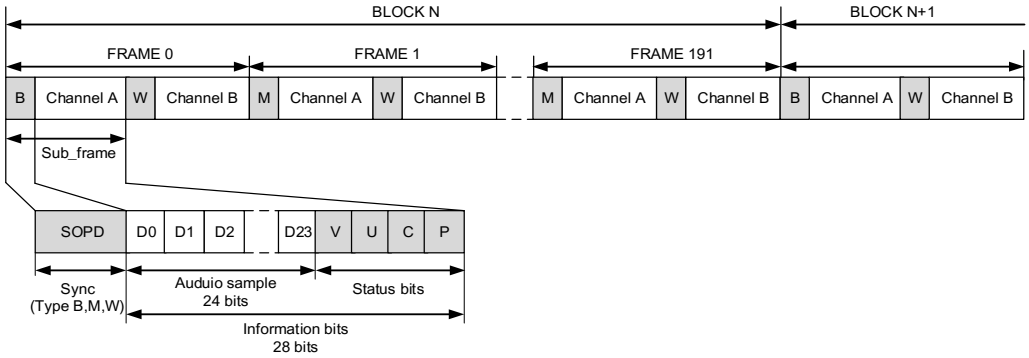
Slot	Name	Description
0	SDI TAG	MSBs indicate witch slots contain valid data
1	STATUS ADDR read port	MSBs echo register address; LSBs indicate which slots request data
2	STATUS DATA read port	16-bit command register read data
3,4	PCM L&R ADC record	16-, 18- or 20-bit PCM data from Left and Right inputs
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

26.3.10. 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 26-15 SPDIF data formate shows SPDIF block formate and sub_frame formate.

Figure 26-15 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 26-5. SOPD mode*

Table 26-5. 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 26-6 Parity bit calculation*.

Table 26-6 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.

26.3.11. 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.

26.3.12. 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.

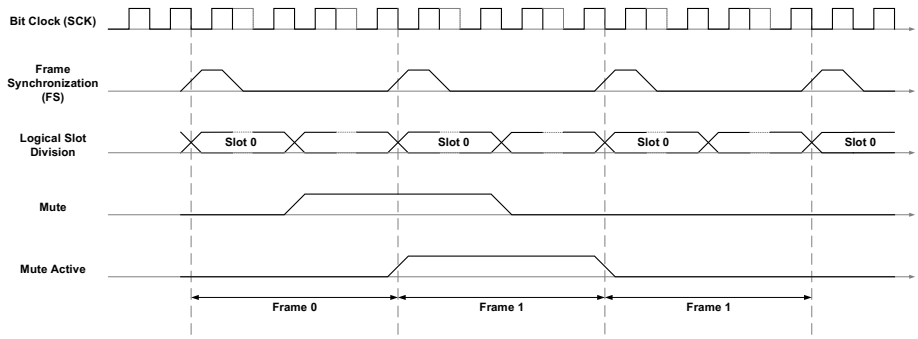
When the SAI audio sub-block is configured as transmitter and mute is configured, data is still read from the FIFO, put into the shift register as usual when mute take effect at next frame, the only difference is that the SD output if forced to a specific value determined by the mute value (MTVAL) configuration 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 26-16 Mute frame activation*.

Figure 26-16 Mute frame activation



SD output behavior under different configuration is summarized in the *Table 26-7 Mute frame composition*.

Table 26-7 Mute frame composition

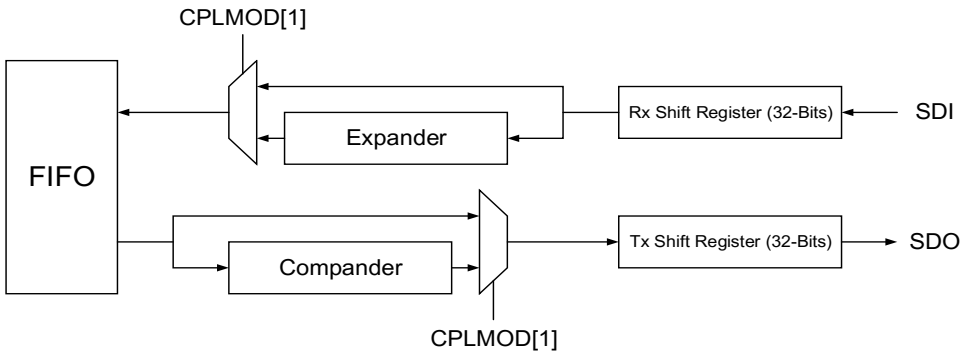
Slot Number	Mute Value = 1	Mute Value = 0
<= 2	Frame before Mute active is repeated on SD Output	Forced to 0
>2	Forced to 0	Forced to 0

26.3.13. 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 26-17 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 26-8 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 26-8 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 26-9 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 26-9 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 26-10 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 26-10 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 26-11 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 26-11 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

26.3.14. 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.

26.3.15. 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.

26.3.16. 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.

26.3.17. 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.

26.3.18. 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 set. 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 (CKERR) 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 (CKERRIE) 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.

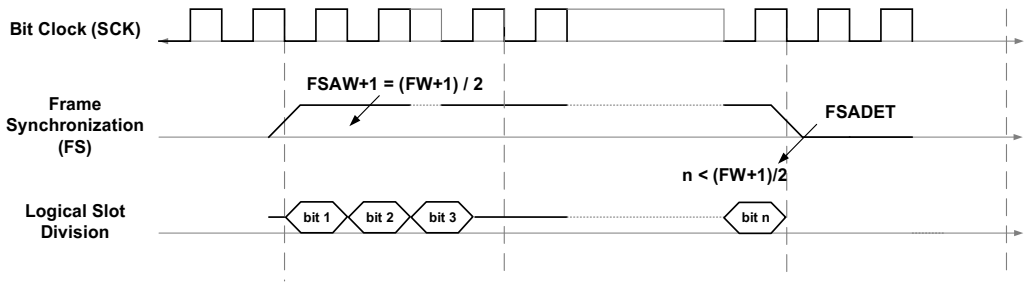
An interrupt is generated when both frame synchronization advanced detected flag (FSADET) in status register SAI_BxSTAT and frame synchronization advanced detection interrupt enable in interrupt enable SAI_BxINTEN register are set.

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 26-18 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.

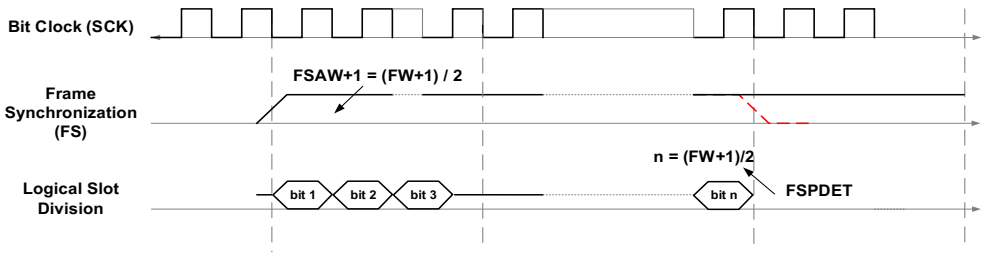
An interrupt is generated when both frame synchronization postpone detected flag (FSPDET)

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 26-19 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.

26.3.19. Interrupts

The *Table 26-12 Interrupt control* summarizes all the interrupt sources present in each audio sub-block.

Table 26-12 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
CKERR	Error	OPTMOD = Master; BYPASS = 1	CKERRIE	CKERRC
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.

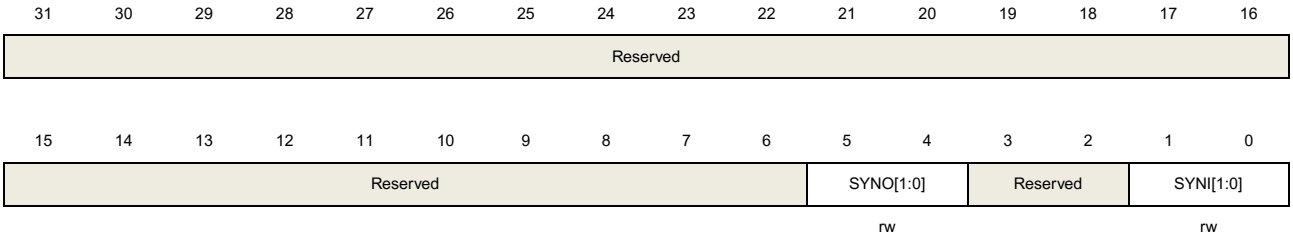
26.4. Register definition

SAI base address: 0x4001 5800

26.4.1. Synchronize configuration register (SAI_SYNCFG)

Address offset: 0x00
Reset value: 0x0000 0000

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

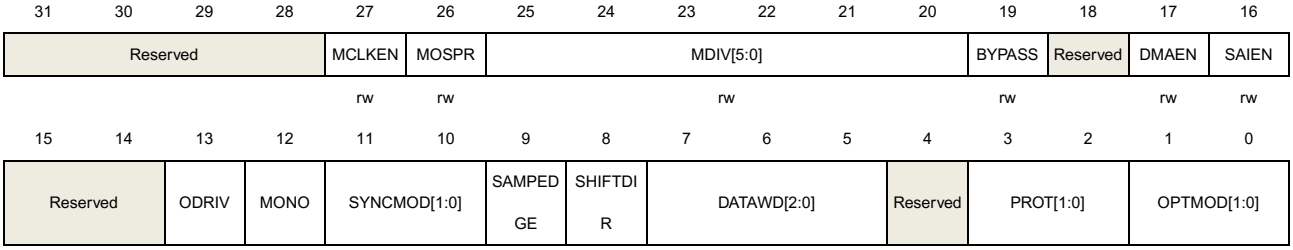


Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5:4	SYNO[1:0]	Synchronization outputs 00: No synchronization output signals. 01: Block 0 used for further synchronization for others SAI 10: Block 1 used for further synchronization for others SAI 11: Reserved. These bits must be set when both audio block (0 and 1) are disabled. 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. 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).

26.4.2. 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).



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 0: The master clock is enable 1: The master clock is enabled independently of SAIEN bit
26	MOSPR	The master clock oversampling rate. 0: MCLK = 256 * F_{fs} 1: MCLK = 512 *F_{fs}
25:20	MDIV[5:0]	Master clock divider ratio. 0000: Primary frequency divider logic bypass, Otherwise, please refer to formula 30-1 in chapter ***Clock divider*** for output frequency calculation by the formula Note: This control field is ineffective when SAI configured in slave mode. Note: This control field has to be set before SAI is enabled.
19	BYPASS	Clock divider logic bypass. 0: Clock divider ratio is applied to both primary and secondary divider logic. 1: Clock divider logic is bypassed.
18	Reserved	Must be kept at reset value.
17	DMAEN	DMA enable. 0: DMA disabled 1: DMA enabled 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. 0: SAI sub-block is disabled. 1: SAI sub-block is enabled, this control field can only be set when SAIEN is 0. 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. Note: SAI slave must be enabled before SAI master.
15:14	Reserved	Must be kept at reset value.
13	ODRIV	Output Drive. 0: SAI sub-block output driven only when SAIEN is set. 1: SAI sub-block output driven according to ODRIV setting. Note: This control field has to be set after SAI configuration but before SAI sub-block enabled.
12	MONO	Stereo and Mono mode selection.

		0: Stereo mode. 1: Mono mode. 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. 00: Asynchronous with the other sub-block. 01: Synchronous with the other sub-block, when this mode is selected, users must configure the operating mode to slave. 10: Synchronous with an external SAI audio sub-block, when this mode is selected, users must configure the operating mode to slave. 11: Reserved. Note: It is required to be configured when audio sub-block is disabled. Note: If the protocol select as SPDIF, the mode should be configured as asynchronous.
9	SAMPEDGE	Sampling clock edge. 0: Data sampled on SCK falling edge. 1: Data sampled on SCK rising edge. Note: This control field is ignored in SPDIF mode. Note: It is required to be configured when audio sub-block is disabled.
8	SHIFTDIR	Shift direction. 0: Data is shifted with MSB first. 1: Data is shifted with LSB first. 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. 000: Reserved. 001: Reserved. 010: 8-bits wide 011: 10-bits wide. 100: 16-bits wide. 101: 20-bits wide. 110: 24-bits wide. 111: 32-bits wide. In compander mode, data width is fix to 8-bits width by the algorithm itself. Note: It is required to be configured when audio sub-block is disabled. Note: This control field is ignored in SPDIF mode. 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.

		00: Polymorphic. 01: SPDIF. 10: AC'97. 10: Reserved. 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. Note: It is required to be configured when audio sub-block is disabled.
1:0	OPTMOD[1:0]	Operating mode. 00: Master transmitter. 01: Master receiver. 10: Slave transmitter. 11: Slave receiver. If the protocol select as SPDIF, the operating mode will be forced to configure as master transmitter. Note: It is required to be configured when audio sub-block is disabled.

26.4.3. 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. 00: No compansion applied. 01: Reserved. 10: Mu-law algorithm. 10: A-law algorithm. 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. Note: Compansion should be applied when TDM protocol is configured.
13	CPLMOD	Complement mode. This control field is used to select the complement option in compansion mode. 0: Data represented in 1's complement form. 1: Data represented in 2's complement form.
12:7	MTFCNT[5:0]	Mute frame count. 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. 0: 0 is sent via the Serial Data (SD) line when mute is on. 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. Note: This control field is meaningful only when the audio sub-block is configured as transmitter, and is ignored in SPDIF mode. 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. 0: Mute mode off 1: Mute mode on 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. Note: Mute will take effect at next frame if mute is set somewhere inside the on-going frame.
4	SDOM	Serial data output management. 0: SD output is driven entirely during the audio frame. 1: SD output is released near inactive slots. 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. 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

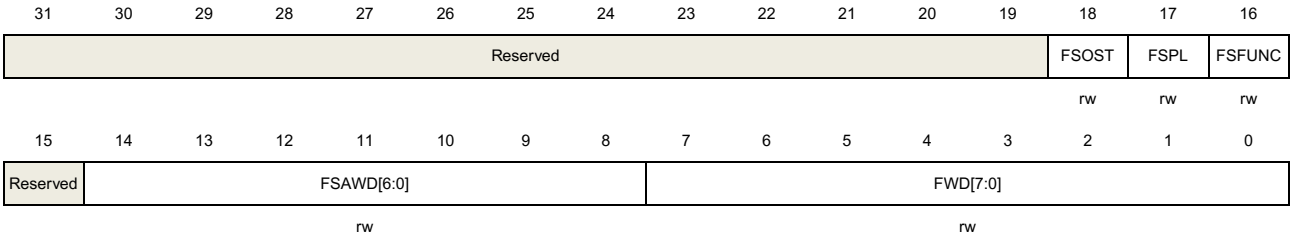
		current frame is the last frame. Note: Whether SD outputs is driven or not on the empty section precede a slot and the offset section followed by a slot is determined by that slot only, if that slot is active, the SD line will be driven, otherwise it will be released.
3	FLUSH	FIFO flush. 0: No FIFO flush. 1: Perform FIFO flush. 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. 000: FIFO empty. 001: FIFO quarter full. 010: FIFO half full. 011: FIFO three quarter full. 100: FIFO full. 101: Reserved. 110: Reserved. 111: Reserved. Note: FIFO threshold is used in conjunction with FIFO status (FFSTAT) control field to determine FIFO request (FFREQ) is CPU and DMA mode.

26.4.4. 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. 0: FS active edge asserted at the beginning of the first bit of the first slot. 1: FS active edge asserted one bit cycle before normal FS when FSOST is 0. 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.

		0: FS low active polarity 1: FS high active polarity. 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. 0: FS only defines frame start. 1: FS define both frame start and channel number. Note: This control field must be configured before the audio sub-block is enabled. 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. 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. Note: This control field must be configured before the audio sub-block is enabled, and it is meaningless in AC'97 mode. 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 Note: This control field must be configured before the audio sub-block is enabled, and it is meaningless in AC'97 mode. 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.

26.4.5. 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).

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

Bits	Fields	Descriptions
31:16	SLOTAV[15:0]	Slot activation vector. 0: Slot inactive. 1: Slot active. 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. Note: This control field must be configured before the audio sub-block is enabled. 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. 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. Note: This control field must be configured before the audio sub-block is enabled. Note: This control field is meaningless in AC'97 mode.
7:6	SLOTWD[1:0]	Slot width. 00: Slot width equals' data width. 01: Slot width of 16-bits. 10: Slot width of 32-bits. 11: Reserved. Slot width must be greater or equal to data width to hold the data, Otherwise the behavior is not guaranteed. Note: This control field must be configured before the audio sub-block is enabled. 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. 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. Note: This control field must be configured before the audio sub-block is enabled. Note: This control field is meaningless in AC'97 mode.

26.4.6. 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. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if FSPDET and FSPDETIE are both set. Note: This control field is meaningless when the audio sub-block is configured as master. Note: This control field is meaningless in AC'97 mode.
5	FSADETIE	Frame synchronization advanced detection interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if FSADET and FSADETIE are both set. Note: This control field is meaningless when the audio sub-block is configured as master. Note: This control field is meaningless in AC'97 mode.
4	ACNRDYIE	Audio codec not ready interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if ACNRDY and ACNRDYIE are both set. Note: This control field has a meaning only when the audio sub-block is configured as receiver. Note: This control field has a meaning only when AC'97 mode is selected.
3	FFREQIE	FIFO request interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if FFREQ and FFREQIE are both set. 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. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if ERRCK and ERRCKIE are both set. 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. Note: This control filed is used when TDM mode is selected, it is meaningless in

		other modes.
1	MTDETIE	Mute detection interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if MTDET and MTDETIE are both set. 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. 0: Interrupt disabled. 1: Interrupt enabled. An interrupt is generated if OUERR and OUERRIE are both set.

26.4.7. 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. 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. In case OPTMOD is configured as receiver: 000: Empty. 001: Empty <FIFO_Level<= 1/4_Full. 010: 1/4_Full <FIFO_Level<= 1/2_Full. 011: 1/2_Full <FIFO_Level<= 3/4_Full. 100: 3/4_Full <FIFO_Level< Full 101: Full. And when OPTMOD is configured as transmitter: 000: Empty. 001: Empty <FIFO_Level< 1/4_Full. 010: 1/4_Full <= FIFO_Level< 1/2_Full. 011: 1/2_Full <= FIFO_Level< 3/4_Full.

		100: 3/4_Full <= FIFO_Level< Full 101: Full.
15:7	Reserved	Must be kept at reset value.
6	FSPDET	Frame synchronization postponed detection. 0: Correct FS edge reception. 1: FS edge reception postponed. FS edge reception postponed could generate an interrupt if FSPDETIE is set. This flag is cleared by FSPDETC control filed. Note: This control field has a meaning only when the audio sub-block is configured as slave.
5	FSADET	Frame synchronization advanced detection. 0: Correct FS edge reception. 1: FS edge reception advanced. FS edge reception advanced could generate an interrupt if FSADETIE is set. This flag is cleared by FSADETC control filed. Note: This control field has a meaning only when the audio sub-block is configured as slave.
4	ACNRDY	Audio codec not ready. 0: AC'97 codec ready. 1: AC'97 codec not ready. 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. AC'97 codec not ready could generate an interrupt if ACNRDYIE is set. This flag is cleared by ACNRDYC control field. Note: This control field is used only in AC'97 mode.
3	FFREQ	FIFO request 0: No FIFO request. 1: FIFO write or read request. FIFO request could generate an interrupt if FFREQIE is set. 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. 0: Correct clock configuration 1: Error clock configuration. Error clock configuration could generate an interrupt if ERRCKIE is set. This flag is cleared by ERRCKC control field. 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. 0: No mute detected

		1: Mute detected. Mute detection could generate an interrupt if MTDETIE is set. This flag is cleared by MTDETC control field. 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. 0: No FIFO overrun or underrun detected 1: FIFO overrun or underrun detected. FIFO overrun or underrun interrupt could be generated if OUERRIE is set. This flag is cleared by OUERRC control field. 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. FIFO underrun is generated only when the audio sub-block is configured as transmitter, the FIFO is empty and a transfer request is present.

26.4.8. 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. Writing 1 clears FSPDET flag. Note: This control field is not used in AC'97 or SPDIF mode. Note: This field always reads as 0.
5	FSADETC	Frame synchronization advanced detection interrupt clear. Writing 1 clears FSADET flag. 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.

		Writing 1 clears ACNRDY flag. Note: This control field is only used in AC'97 mode. Note: This field always reads as 0.
3	Reserved	Must be kept at reset value.
2	ERRCKC	Clock error interrupt clear. Writing 1 clears ERRCK flag. Note: This control field is only used when the audio sub-block is configured as master and BYPASS is set to 0. Note: This field always reads as 0.
1	MTDETC	Mute detection interrupt clear. Writing 1 clears MTDET flag. Note: This field always reads as 0.
0	OUERRC	Overrun or underrun interrupt clear. Writing 1 clears OUERR flag. Note: This field always reads as 0.

26.4.9. 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. Write and read operations are performed on the FIFO directly.

27. Digital camera interface (DCI)

27.1. 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.

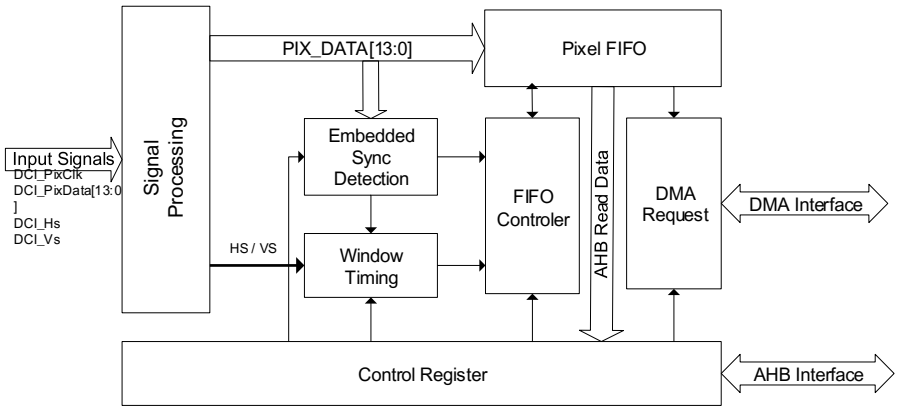
27.2. Characteristics

- 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
- JPEG compression format supported
- Hard/embedded synchronous signals supported

27.3. Block diagram

The DCI contains these modules: Signal Processing, Pixel FIFO, FIFO controller, window timing, embedded sync detection, DMA interface and control register.

Figure 27-1. 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.

27.4. Signal description

Table 27-1. PINs used by DCI

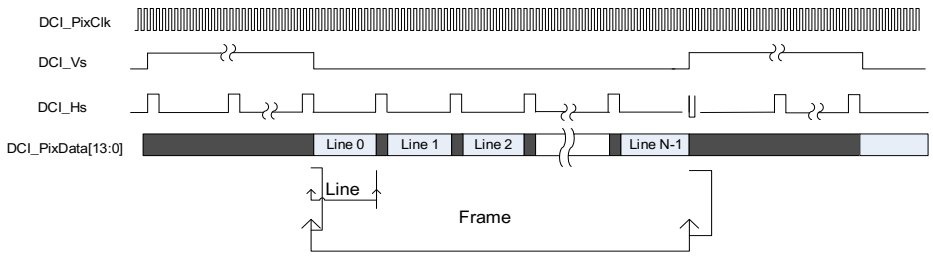
Direction	Pins	Signals	Width	Description
Input	DCI_PIXCLK	DCI_PixClk	1	DCI Pixel Clock
Input	DCI_D	DCI_PixData	14	DCI Pixel Data
Input	DCI_HSYNC	DCI_HSYNC	1	DCI Horizontal Synchronization
Input	DCI_VSYNC	DCI_VSYNC	1	DCI Vertical Synchronization

27.5. Function overview

27.5.1. 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_D[13:0] at rising or falling edge of DCI_PIXCLK (clock polarity is configured by CKS bit in DCI_CTL).

Figure 27-2. Hardware synchronization mode



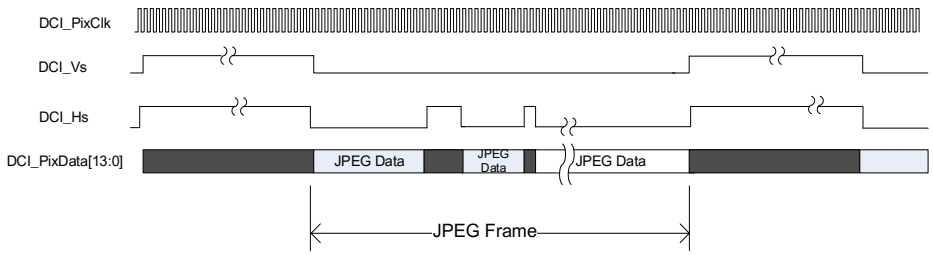
The above figure assumes that the polarities of both DCI_HSYNC and DCI_VSYNC are high during blanking period, so the data on DCI_D lines is only valid when both DCI_HSYNC and DCI_VSYNC are low.

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 27-3. Hardware synchronization mode: JPEG format supporting



27.5.2. Embedded synchronization mode

DCI supports embedded synchronization mode. In this mode there are only DCI_D 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-MN, and MN 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 MN byte in FF_00_00_MN 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.

27.5.3. 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.

27.5.4. 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.

27.5.5. 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 27-2. 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 27-3. 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]

27.6. 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 into DCI_INTC register.

Table 27-4. Status/Error flags

Status Flag Name	Description
ELF	End of Line Flag
EFF	End of Frame Flag
OVRF	FIFO Overrun Flag
VSF	Frame VS Blank Flag
ESEF	Embedded Sync Error Flag

27.7. Register definition

DCI base address: 0x5005 0000

27.7.1. 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DCIEN	Reserved		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

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	DCIEN	DCI Enable 0: DCI is disabled 1: DCI is enabled
13:12	Reserved	Must keep the reset value
11:10	DCIF[1:0]	Digital Camera Interface Format 00: 8-bit data on every pixel clock 01: 10-bit data on every pixel clock 10: 12-bit data on every pixel clock 11: 14-bit data on every pixel clock
9:8	FR[1:0]	Frame Rate FR defines the frame capture rate in continuous capture mode 00: Capture all frames 01: Capture one in 2 frames 10: Capture one in 4 frames 11: Reserved
7	VPS	Vertical Polarity Selection 0: Low level during blanking period 1: High level during blanking period
6	HPS	Horizontal Polarity Selection 0: Low level during blanking period 1: High level during blanking period

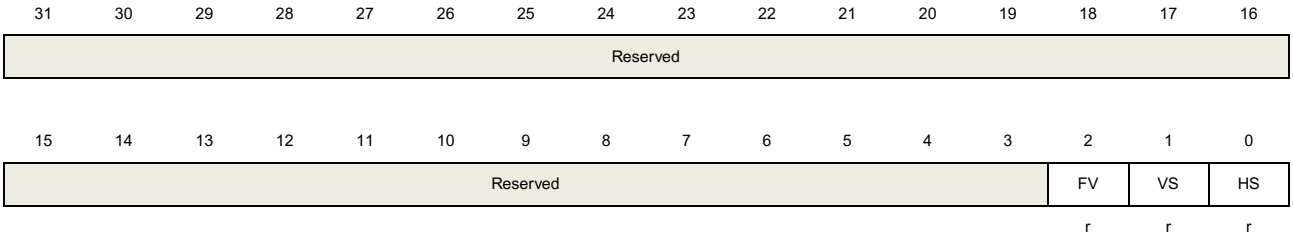
5	CKS	Clock Polarity Selection 0: Capture at falling edge 1: Capture at rising edge
4	ESM	Embedded Synchronous Mode 0: Embedded synchronous mode is disabled 1: Embedded synchronous mode is enabled
3	JM	JPEG Mode 0: JPEG mode is disabled 1: JPEG mode is enabled
2	WDEN	Window Enable 0: Window is disabled 1: Window is enabled
1	SNAP	Snapshot Mode 0: Continuous capture mode 1: Snapshot capture mode
0	CAP	Capture Enable 0: Frame not captured 1: Frame is captured

27.7.2. Status register0 (DCI_STAT0)

Address offset: 0x04

Reset value: 0x0000 0003

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



Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	FV	FIFO Valid 0: No valid pixel data in FIFO 1: Valid pixel data in FIFO
1	VS	VS line status 0: Not in vertical blanking period 1: In vertical blanking period

0	HS	HS line status 0: Not in horizontal blanking period 1: In horizontal blanking period

27.7.3. 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											ELF	VSF	ESEF	OVRF	EFF
											r	r	r	r	r

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	ELF	End of Line Flag 0: No end of line flag 1: A line is captured by DCI
3	VSF	Vsync Flag 0: No vsync flag 1: A vsync blanking detected
2	ESEF	Embedded Synchronous Error Flag 0: No embedded synchronous error flag 1: A embedded synchronous error detected
1	OVRF	FIFO Overrun Flag 0: No FIFO overrun 1: A FIFO overrun occurs
0	EFF	End of Frame Flag 0: No end of frame flag 1: A frame is captured by DCI

27.7.4. 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											ELIE	VSIE	ESEIE	OVRIE	EFIE
											rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	ELIE	End of Line Interrupt Enable 0: End of line flag won't generate interrupt 1: End of line flag will generate interrupt
3	VSIE	Vsync Interrupt Enable 0: Vsync flag won't generate interrupt 1: Vsync flag will generate interrupt
2	ESEIE	Embedded Synchronous Error Interrupt Enable 0: Embedded synchronous error flag won't generate interrupt 1: Embedded synchronous error flag will generate interrupt
1	OVRIE	FIFO Overrun Interrupt Enable 0: FIFO overrun won't generate interrupt 1: FIFO overrun will generate interrupt
0	EFIE	End of Frame Interrupt Enable 0: End of frame flag won't generate interrupt 1: End of frame flag will generate interrupt

27.7.5. 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ELIF	VSIF	ESEIF	OVRIF	EFIF
											r	r	r	r	r

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.

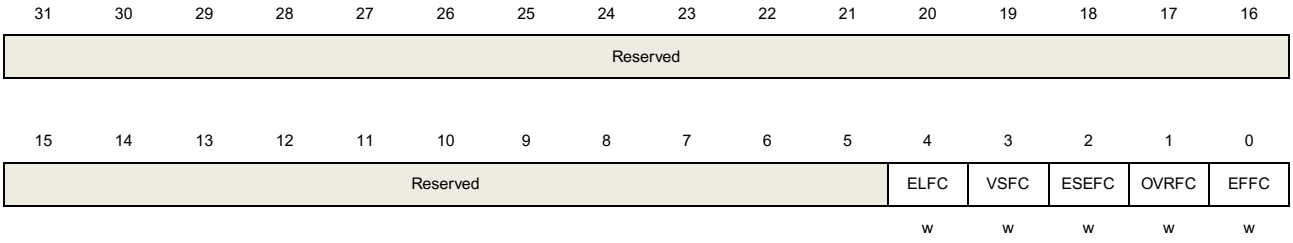
4	ELIF	End of line interrupt flag
3	VSIF	Vsync interrupt flag
2	ESEIF	Embedded synchronous error interrupt flag
1	OVRIF	FIFO overrun interrupt flag
0	EFIF	End of frame interrupt flag

27.7.6. **Interrupt** flag clear register (DCI_INTC)

Address offset: 0x14

Reset value: 0x0000 0000

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



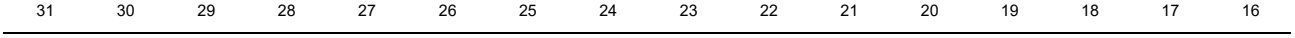
Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	ELFC	End of Line Flag Clear Write 1 to clear end of line flag
3	VSFC	Vsync flag clear Write 1 to clear vsync flag
2	ESEFC	Clear embedded synchronous Error Flag Write 1 to clear embedded synchronous error flag
1	OVRFC	Clear FIFO Overrun Flag Write 1 to clear FIFO overrun flag
0	EFFC	Clear End of Frame Flag Write 1 to clear end of frame flag

27.7.7. **Synchronization codes register (DCI_SC)**

Address offset: 0x18

Reset value: 0x0000 0000

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



FE[7:0]	LE[7:0]
rw	rw

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

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

27.7.8. 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

27.7.9. 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 Zero means the first line
15:14	Reserved	Must keep the reset value
13:0	WHSP[13:0]	Window Horizontal Start Position Zero means the first pixel clock in a line

27.7.10. 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
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 WVSZ=x means x+1 lines
15:14	Reserved	Must be kept at reset value
13:0	WHSZ[13:0]	Window Horizontal Size WHSZ=x means x+1 pixels clock in a line

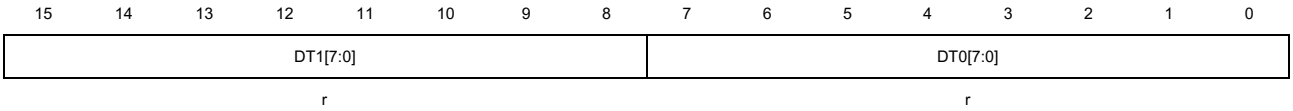
27.7.11. DATA register (DCI_DATA)

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							



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

28. TFT-LCD interface (TLI)

28.1. 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.

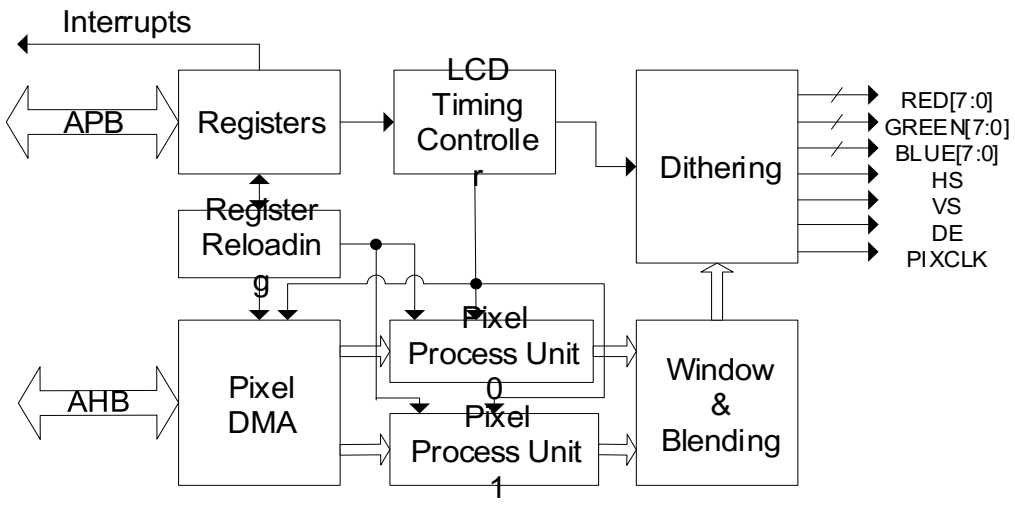
28.2. Characteristics

- 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

28.3. Block diagram

Figure below 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 AHB clock and fetches pixel data from system memory using AHB bus. The remaining modules work in TLI clock. The TLI clock is divided from PLLSAI-R clock. The parameters of PLLSAI and dividing factor are configured in RCU module.

Figure 28-1. TLI module block diagram



28.4. Signal description

TLI provides a 24-bit RGB Parallel display interface, which is shown in table below.

Table 28-1. Pins of display interface provided by TLI

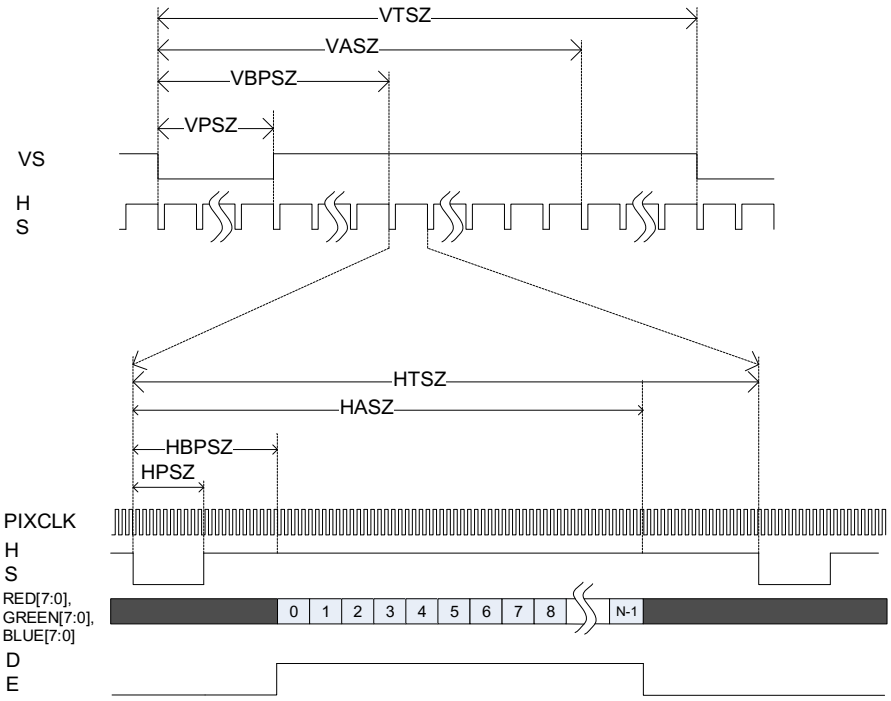
Direction	Name	Width	Description
Output	HS	1	Horizontal Synchronous
Output	VS	1	Vertical Synchronous
Output	DE	1	Data Enable
Output	PIXCLK	1	Pixel Clock
Output	RED[7:0]	8	Pixel Red Data
Output	GREEN[7:0]	8	Pixel Green Data
Output	BLUE[7:0]	8	Pixel Blue Data

28.5. Function overview

28.5.1. LCD display timing

LCD interface is a synchronous data interface with pixel clock, pixel data and horizontal and vertical synchronous signals. The figure below 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 28-2. Display timing diagram



28.5.2. 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 AHB transaction to fetches 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 AHB 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+3.

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.

28.5.3. Pixel formats

The Pixel DMA pushes pixel data into PPU in 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 the table below. 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 28-2. Supported pixel formats

PPF[2:0]	Pixel Format
000	ARGB8888
001	RGB888
010	RGB565
011	ARGB1555
100	ARGB4444
111	AL88
101	L8
110	AL44

28.5.4. Layer window and blending function

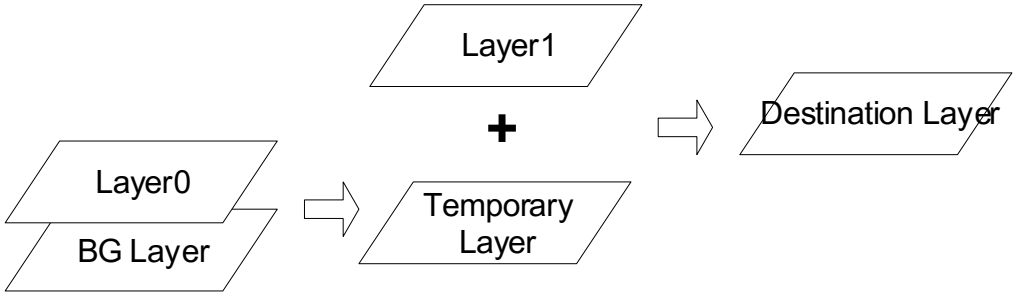
TLI supports window function for each layer and blending function between two layers. TLI first perform window operation to each layer and then blend two layers into a frame.

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 28-3. Block diagram of Blending



Blending formula

The general blending formula is:

$$BC = BF_1 \times C + BF_2 \times C_S \quad (28\text{-}1)$$

- 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.

28.5.5. 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, then 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 application set 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.

28.5.6. 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.

28.6. 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 28-3. Status flags

Status Flag Name	Description
LMF	Line Mark Flag
LCRF	Layer Configuration Reloaded Flag

Table 28-4. Error flags

Error Flag Name	Description
TEF	Transaction Error Flag
FEF	FIFO Error Flag

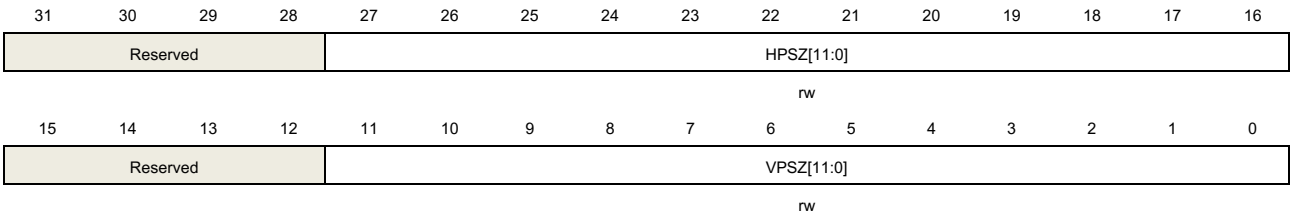
28.7. Register definition

TLI base address: 0x4001 6800

28.7.1. Synchronous pulse size register (TLI_SPSZ)

Address offset: 0x08
Reset value: 0x0000 0000

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

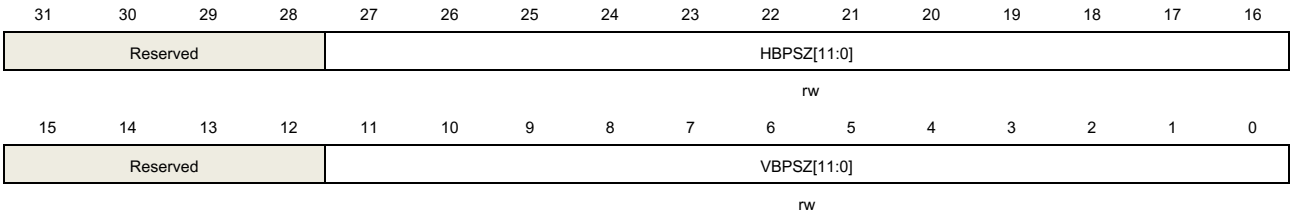


Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:16	HPSZ[11:0]	Size of the horizontal synchronous pulse The HPSZ value should be configured to the pixels number of horizontal synchronous pulse minus 1.
15:12	Reserved	Must keep the reset value
11:0	VPSZ[11:0]	Size of the vertical synchronous pulse The VPSZ value should be configured to the pixels number of vertical synchronous pulse minus 1.

28.7.2. Back-porch size register (TLI_BPSZ)

Address offset: 0x0C
Reset value: 0x0000 0000

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



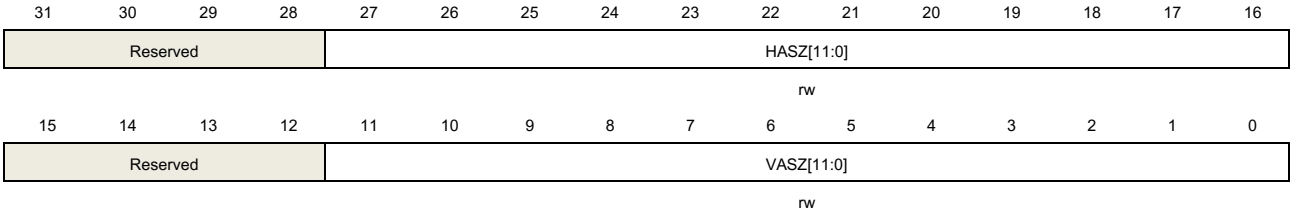
Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:16	HBPSZ[11:0]	Size of the horizontal back porch plus synchronous pulse

		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 The VBPSZ value should be configured to the pixels number of vertical back porch and synchronous pulse minus 1.

28.7.3. Active size register (TLI_ASZ)

Address offset: 0x10
Reset value: 0x0000 0000

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

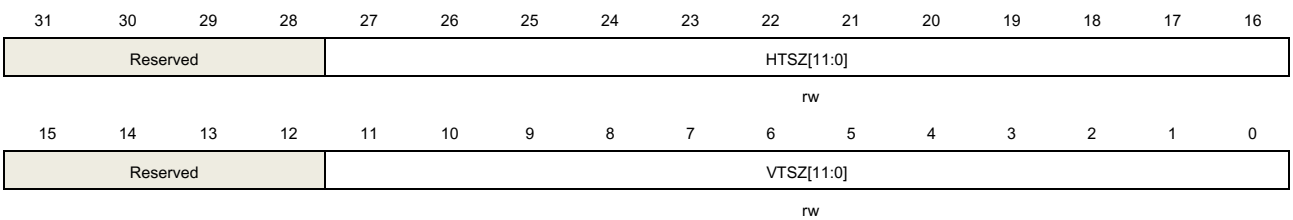


Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:16	HASZ[11:0]	Size of the horizontal active area width plus back porch and synchronous pulse The HASZ value should be configured to the pixels number of horizontal active area width plus back porch and synchronous pulse minus 1.
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 The VASZ value should be configured to the pixels number of vertical active area height plus back porch and synchronous pulse minus 1.

28.7.4. 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 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 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.

28.7.5. 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 0: Horizontal Synchronous Pulse active low 1: Horizontal Synchronous Pulse active high
30	VPPS	Vertical pulse polarity selection 0: Vertical Synchronous Pulse active low 1: Vertical Synchronous Pulse active high
29	DEPS	Data enable polarity selection 0: Data Enable active low 1: Data Enable active high
28	CLKPS	Pixel clock polarity selection 0: Pixel Clock is TLI clock 1: Pixel Clock is inverted TLI clock
27:17	Reserved	Must be kept at reset value.
16	DFEN	Dither function enable 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 Fixed to 2, read only
11	Reserved	Must be kept at reset value.
10:8	GDB[2:0]	Green channel dither bits number Fixed to 2, read only
7	Reserved	Must be kept at reset value.
6:4	BDB[2:0]	Blue channel dither bits number Fixed to 2, read only
3:1	Reserved	Must be kept at reset value.
0	TLIEN	TLI enable bit 0: TLI disable 1: TLI enable

28.7.6. Reload layer register (TLI_RL)

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														FBR	RQR
														rw	rw

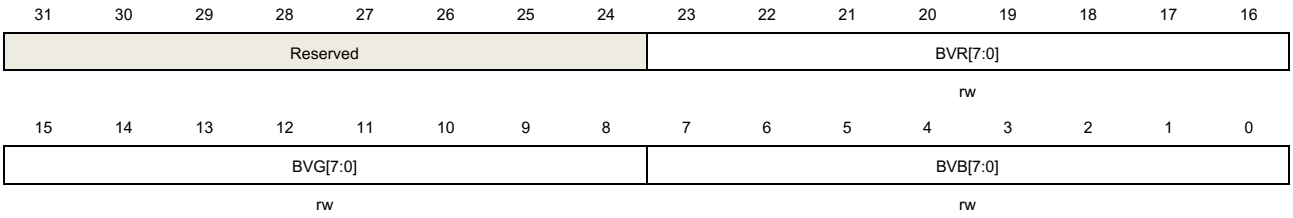
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	FBR	Frame blank reload This bit is set by software and cleared by hardware after reloading 0: Reload disable 1: The layer configuration will be reloaded into core at frame blank
0	RQR	Request reload This bit is set by software and cleared by hardware after reloading 0: Reload disable 1: The layer configuration will be reloaded into core after this bit sets

28.7.7. Background color register (TLI_BGC)

Address offset: 0x2C

Reset value: 0x0000 0000

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



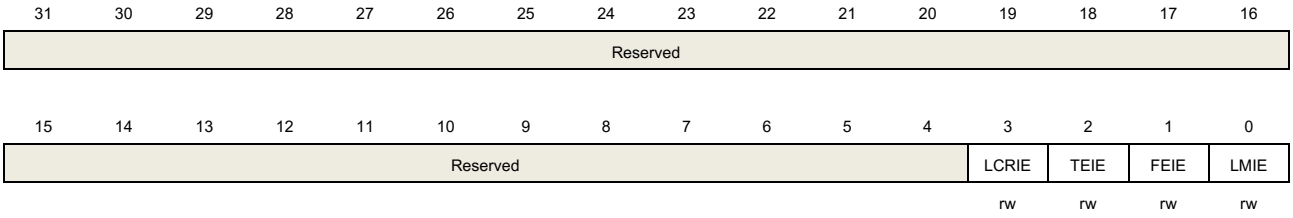
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:16	BVR[7:0]	Background value red
15:8	BVG[7:0]	Background value green
7:0	BVB[7:0]	Background value blue

28.7.8. 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 0: Layer configuration reloaded flag won't generate an interrupt 1: Layer configuration reloaded flag will generate an interrupt
2	TEIE	Transaction error interrupt enable 0: Transaction error flag won't generate an interrupt 1: Transaction error flag will generate an interrupt
1	FEIE	FIFO error interrupt enable 0: FIFO error flag won't generate an interrupt 1: FIFO error flag will generate an interrupt

0	LMIE	Line mark interrupt enable 0: Line mark flag won't generate an interrupt 1: Line mark flag will generate an interrupt

28.7.9. 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 0: No layer configuration reloaded flag 1: Layer configuration is reloaded triggered by FBR bit in TLI_RL
2	TEF	Transaction error flag 0: No transaction error flag 1: A transaction error on AHB bus occurs
1	FEF	FIFO error flag 0: No FIFO error flag 1: A FIFO under-run error occurs The under-run error occurs when the value written in TLI_LxFLLEN and TLI_LxFTLN is less than required.
0	LMF	Line mark flag 0: No line mark flag 1: Line number reaches the specified value in TLI_LM

28.7.10. 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 Write 1 to clear layer configuration reloaded flag
2	TEC	Transaction error flag clear Write 1 to clear transaction error flag
1	FEC	FIFO error flag clear Write 1 to clear FIFO error flag
0	LMC	Line mark flag clear Write 1 to clear line mark flag

28.7.11. 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 The LMF bit in TLI_INTF will be set after the line number reaches this value

28.7.12. 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 Horizontal position of the current displayed pixel
15:0	VPOS[15:0]	Vertical position Vertical position of the current displayed pixel

28.7.13. Status register (TLI_STAT)

Address offset: 0x48

Reset value: 0x0000 000F

This register 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												HS	VS	HDE	VDE
												r	r	r	r

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 0: HPOS in TLI_CPPOS register is not between the HBPSZ in TLI_BPSZ register and HASZ in TLI_ASZ register. 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 0: VPOS in TLI_CPPOS register is not between the VBPSZ in TLI_BPSZ register and VASZ in TLI_ASZ register. 1: VPOS in TLI_CPPOS register is between the VBPSZ in TLI_BPSZ register and VASZ in TLI_ASZ register.

28.7.14. 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).

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											LUTEN	Reserved		CKEYEN	LEN
											rw			rw	rw

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	LUTEN	LUT enable 0: LUT is disabled 1: LUT is enabled
3:2	Reserved	Must be kept at reset value.
1	CKEYEN	Color keying enable 0: Color keying is disabled 1: Color keying is enabled
0	LEN	Layer enable 0: This layer is disabled 1: This layer is enabled

28.7.15. 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

28.7.16. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				WBP[11:0]											
				rw											

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

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

28.7.17. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CKEYR[7:0]							
								rw							

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

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.

28.7.18. 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).

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														PPF[2:0]	
														rw	

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	PPF[2:0]	Packeted Pixel Format These bits configures the Packeted Pixel format 000: ARGB8888 001: RGB888 010: RGB565 011: ARGB1555 100: ARGB4444 101: L8 110: AL44 111: AL88

28.7.19. 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 The Alpha value used for blending

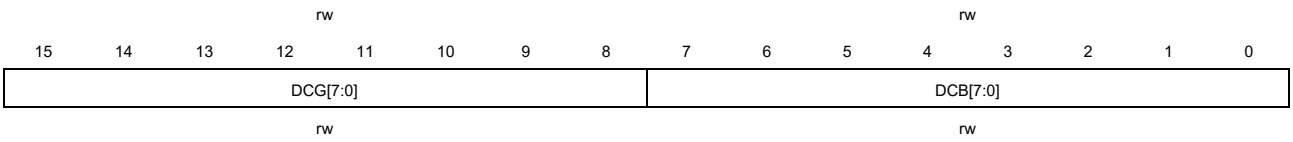
28.7.20. Layer x default color register (TLI_LxDC) (x = 0, 1)

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]							



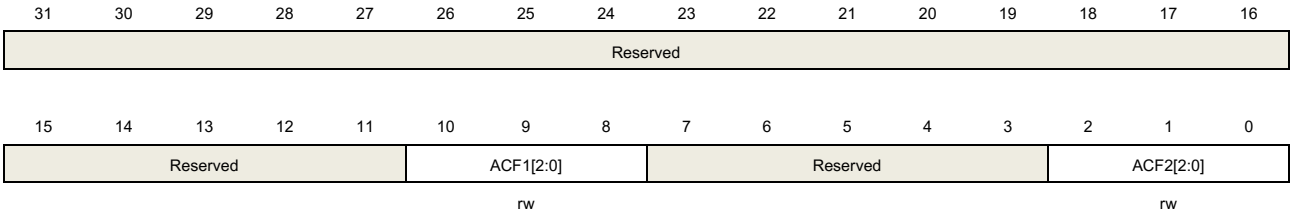
Bits	Fields	Descriptions
31:24	DCA[7:0]	The Default Color ALPHA
23:16	DCR[7:0]	The Default Color Red
15:8	DCG[7:0]	The Default Color Green
7:0	DCB[7:0]	The Default Color Blue

The default color of a layer takes effect when the layer is disabled or outside the window defined in TLI_LxHPOS and TLI_LxVPOS.

28.7.21. 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 000: Reserved 001: Reserved 010: Reserved 011: Reserved 100: normalization Specified Alpha 101: Reserved 110: normalization Pixel Alpha x normalization Specified Alpha 111:Reserved
7:3	Reserved	Must be kept at reset value.
2:0	ACF2[2:0]	Alpha Calculation Factor 2 of Blending Method 000: Reserved 001: Reserved 010: Reserved

011: Reserved

100: Reserved

101:1- normalization Specified Alpha

110: Reserved

111: 1-normalization Pixel Alpha x normalization Specified Alpha

28.7.22. 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 The base address of frame buffer

28.7.23. 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													

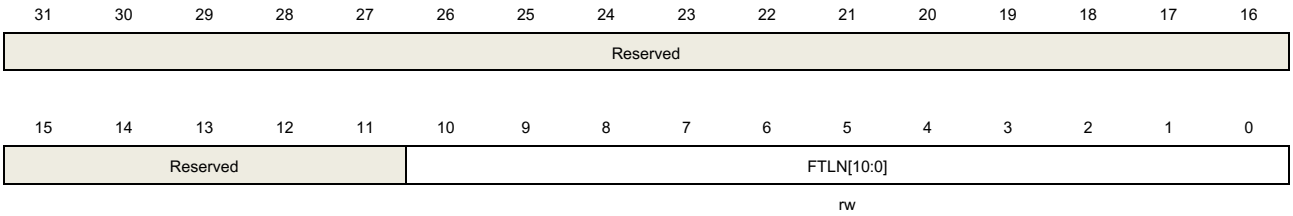
Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:16	STDOFF[13:0]	Frame buffer stride offset This value defines the bytes number from start of a line to the start of next line
15:14	Reserved	Must be kept at reset value.
13:0	FLL[13:0]	Frame line length This value defines the bytes number of a line plus 3

28.7.24. 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).



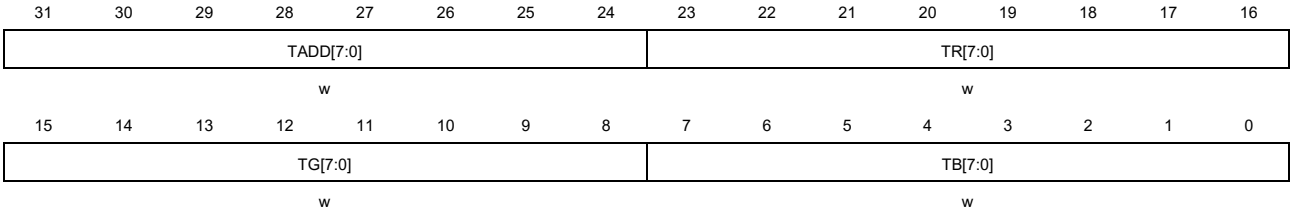
Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10:0	FTLN[10:0]	Frame total line number This value defines the line number in a frame

28.7.25. 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 The entry at this address in LUT will be updated with the value of RED, GREEN and BLUE 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

29. Secure digital input/output interface (SDIO)

29.1. Introduction

The secure digital input/output interface (SDIO) defines the SD/SD I/O /MMC CE-ATA card host interface, which provides command/data transfer between the APB2 system bus and SD memory cards, SD I/O cards, Multimedia Card (MMC), and CE-ATA devices.

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 Multimedia Card system specifications are defined through the Multimedia Card Association website at **www.jedec.org**, published by the JEDEC SOLID STATE TECHNOLOGY ASSOCIATION.

The supported CE-ATA system specifications are defined through the CE-ATA workgroup website at **www.ce-ata.org**.

29.2. Main features

The SDIO features include the following:

- **MMC:** Full support for Multimedia Card System Specification Version 4.2(and previous versions) Card and three different data bus modes: 1-bit (default), 4-bit and 8-bit
- **SD Card:** Full support for *SD Memory Card Specifications Version 2.0*
- **SD I/O:** Full support for *SD I/O Card Specification Version 2.0* card and two different data bus modes: 1-bit (default) and 4-bit
- **CE-ATA:** Full compliance with *CE-ATA digital protocol Version 1.1*
- 48MHz data transfer frequency and 8-bit data transfer mode.
- Interrupt and DMA request to processor.
- Completion Signal enables and disable feature (CE-ATA).

Note: SDIO supports only one SD, SD I/O, MMC4.2 card or CE-ATA device at any one time and a stack of MMC4.1 or previous.

29.3. 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 ***Card functional description***. One data transfer is a bus operation.

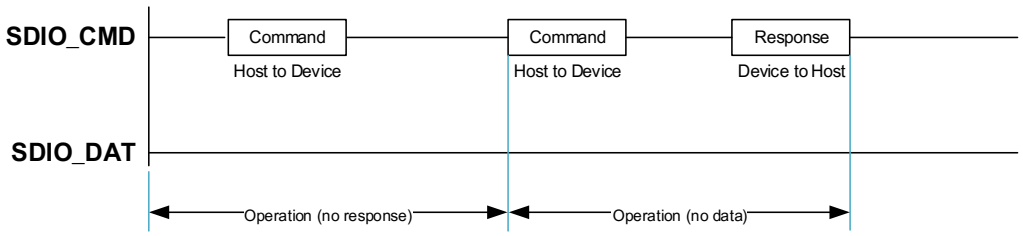
There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. The bits on the DAT0-DAT7 and CMD lines are transferred synchronous to the host clock.

Two 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 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.

The basic transaction on the bus is the command/response transaction (refer to ***Figure 29-1. 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/Device are done in blocks.

Figure 29-1. 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 29-2. SDIO multiple blocks read operation*** is the multiple blocks read operation and ***Figure 29-3. 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. CE-ATA device

has an optional busy before it is ready to receive the data.

Figure 29-2. SDIO multiple blocks read operation

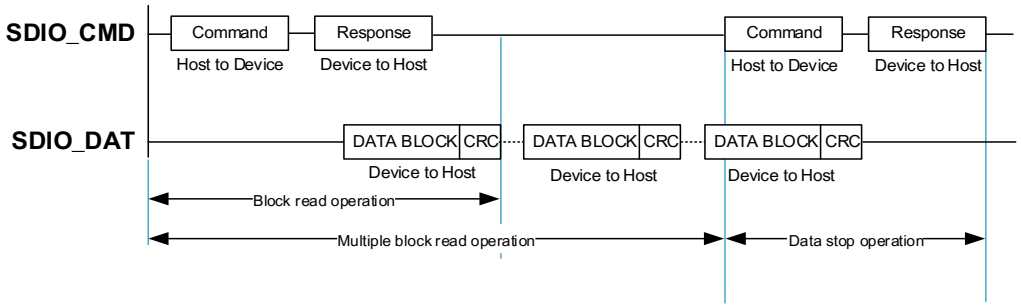
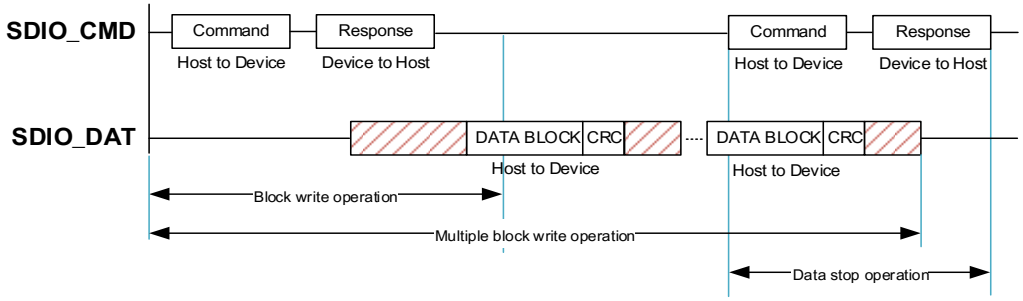


Figure 29-3. SDIO multiple blocks write operation



Data transfers to/from SD memory cards, SD I/O cards (both IO only card and combo card) and CE-ATA device are done in data blocks. Data transfers to/from MMC are done in data blocks or streams. *Figure 29-4. SDIO sequential read operation* and *Figure 29-5. SDIO sequential write operation* are the stream read and write operation.

Figure 29-4. SDIO sequential read operation

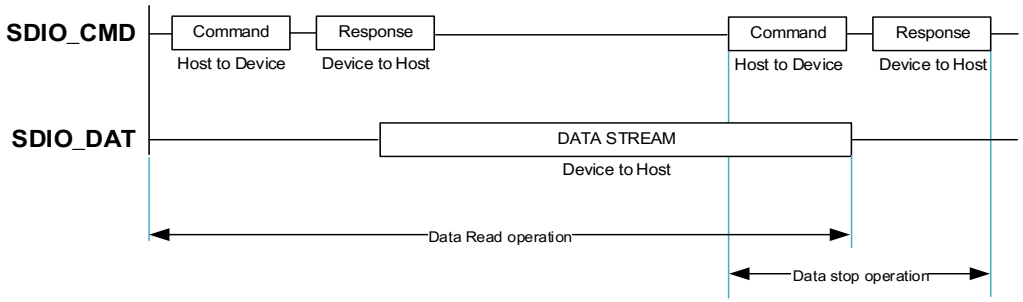
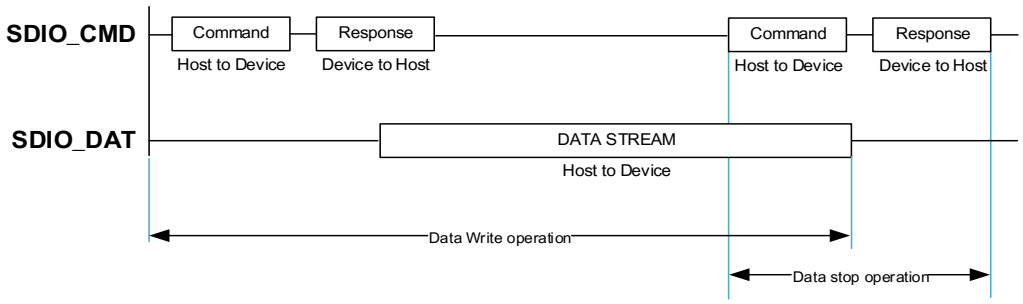


Figure 29-5. SDIO sequential write operation

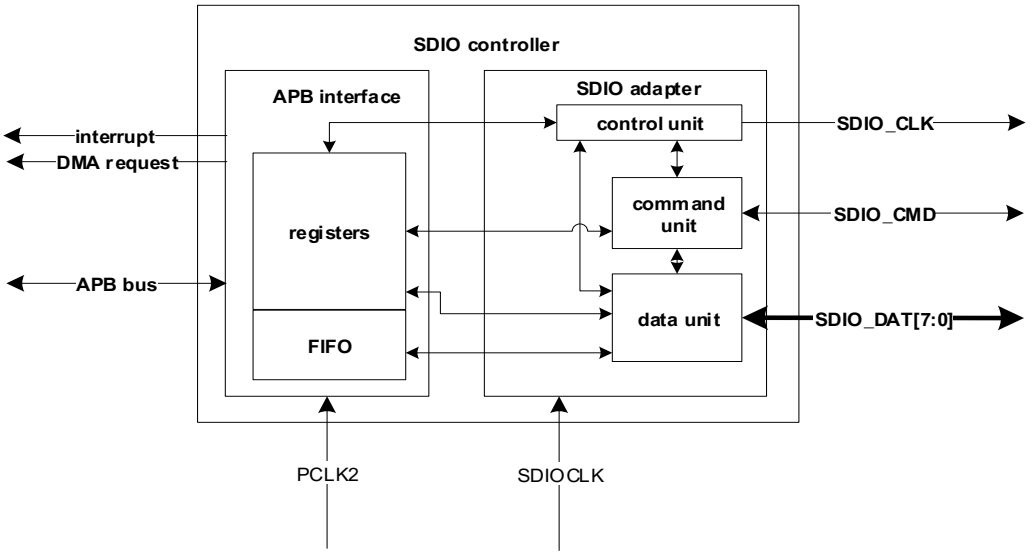


29.4. SDIO functional description

The following figure shows the SDIO structure. There have two 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 APB interface block contains access registers by APB2 bus, contains FIFO unit which is data FIFO used for data transfer, and generates interrupt and DMA request signals.

Figure 29-6. SDIO block diagram



29.4.1. 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) and on all the data lines (SDIO_DAT). The SDIO_CLK frequency can vary between 0 MHz and 20 MHz for a Multimedia Card V3.31,

between 0 and 48 MHz for a Multimedia Card V4.2, or between 0 and 25 MHz for an SD/SD I/O card.

The SDIO uses two clock signals: SDIO adapter clock (SDIOCLK <= 48MHz) and APB2 bus clock (PCLK2).

The frequency of PCLK2 must be no less than the 3/8 frequency of SDIO_CLK.

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 (only for MMC3.31 or previous), and push-pull for command transfer (SD/SD I/O card MMC4.2 use push-pull drivers also for initialization).

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 MMC4.2), 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 and DAT2 (DAT3 internal pull-up is left connected due to the SPI mode CS usage). Correspondingly right after entering to the 8-bit mode the card disconnects the internal pull-ups of lines DAT1, DAT2 and DAT4-DAT7.

Table 29-1. SDIO I/O definitions

Pin function	Direction	Description
SDIO_CLK	O	SD/SD I/O /MMC clock
SDIO_CMD	I/O	Command input/output
SDIO_DAT[7:0]	I/O	Data input/output for data lines DAT[7:0]

The SDIO adapter is an interface to SD/SD I/O /MMC/CE-ATA. It consists of three subunits:

Control unit

The control unit contains the power management functions and the clock management functions for the memory card clock. The power management is controlled by SDIO_PWRCTL register which implements power off or power on. The power saving mode configured by setting CLKPWRSAV bit in SDIO_CLKCTL register, which implements close the SDIO_CLK when the bus is idle. The clock management generates SDIO_CLK to card. The SDIO_CLK is generated by a divider of SDIOCLK when CLKBYP bit in SDIO_CLKCTL register is 0, or directly SDIOCLK when CLKBYP bit in SDIO_CLKCTL register is 1.

The Hardware clock control is enabled by setting HWCLKEN in SDIO_CLKCTL register. This functionality is used to avoid FIFO underrun and overrun errors by hardware control the SDIO_CLK on/off depending on the system bus is very busy or not. When the FIFO cannot receive or transmit data, the host will stop the SDIO_CLK and freeze SDIO state machines to avoid the corresponded error. Only state machines are frozen, the APB2 interface is still alive. So, the FIFO can access by APB2 bus.

Command unit

The command unit implements command transfer to the card. 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 bits 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 0b00/0b10. 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 and WAITDEND disabled	->	CS_Send
3.CSM disabled	->	CS_Idle
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

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.Response Received in CE-ATA mode and interrupt disabled and wait for CE-ATA Command	->	CS_Waitcompl

Completion signal enabled		
2.Response Received in CE-ATA mode and interrupt disabled and wait for CE-ATA Command Completion signal disabled	->	CS_Pend
3.CSM disabled	->	CS_Idle
4.Response received	->	CS_Idle
5.Command CRC failed	->	CS_Idle

CS_Waitcompl	Wait for the Command Completion signal.	
1.CE-ATA Command Completion signal received	->	CS_Idle
2.CSM disabled	->	CS_Idle
3.Command CRC failed	->	CS_Idle

Data unit

The data unit performs data transfers to and from cards. The data transfer uses SDIO_DAT[7:0] signals when 8-bits data width (BUSMODE bits in SDIO_CLKCTL register is 0b10), use SDIO_DAT[3:0] signals when 4-bits 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.

Data state machine

DS_Idle	The data unit is inactive, waiting for send and receive.	
1.DSM enabled and data transfer direction is from host to card	->	DS_WaitS
2.DSM enabled and data transfer direction is from card to host	->	DS_WaitR
3.DSM enabled and Read Wait Started and SD I/O mode enabled	->	DS_Readwait

DS_WaitS	Wait until the data FIFO empty flag is deasserted or data transfer ended.	
1.Data transfer ended	->	DS_Idle
2.DSM disabled	->	DS_Idle
3.Data FIFO empty flag is deasserted	->	DS_Send

DS_Send	Transmit data to the card.	
1.Data block transmitted	->	DS_Busy

2.DSM disabled	->	DS_Idle
3.Data FIFO underrun error occurs	->	DS_Idle
4. Internal CRC error	->	DS_Idle

DS_Busy	Waits for the CRC status flag.	
1.Receive a positive CRC status	->	DS_WaitS
2.Receive a negative CRC status	->	DS_Idle
3.DSM disabled	->	DS_Idle
4.Timeout occurs	->	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.Data timeout reached	->	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	->	DS_WaitR
2.Data transfer ended	->	DS_WaitR
3.Data FIFO overrun error occurs	->	DS_Idle
4.Data received and Read Wait Started and SD I/O mode enabled	->	DS_Readwait
5.DSM disabled or CRC fails	->	DS_Idle

DS_Readwait	Wait for the read wait stop command.	
1.ReadWait stop enabled	->	DS_WaitR
2.DSM disabled	->	DS_Idle

29.4.2. APB2 interface

The APB2 interface implements access to SDIO registers, data FIFO and generates interrupt and DMA request. It includes a data FIFO unit, registers unit, and the interrupt / DMA logic.

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 DMA interface provides a method for fast data transfers between the SDIO data FIFO and memory. The following example describes how to implement this method:

1. Completes the card identification process

2. Increase the SDIO_CLK frequency

3. Send CMD7 to select the card and configure the bus width

4. Configure the DMA1 as follows:

Open the DMA1 controller and clear any pending flags. Configure the DMA1_Channel3 or DMA1_Channel6 Peripheral4 source address register with the memory base address and DMA1_Channel3 or DMA1_Channel6 Peripheral4 destination address register with the SDIO_FIFO register address. Configure DMA1_Channel3 or DMA1_Channel6 Peripheral4 control register (memory with increment transfer, peripheral with not increment transfer, peripheral and memory data size is word size). Program the incremental burst transfer to 4 on peripheral side in DMA1_Channel 3 or DMA1_Channel 6 Peripheral4.

5. Write block to card as follows:

Write the data size in bytes in the SDIO_DATALEN register. Write the block size in bytes (BLKSZ) in the SDIO_DATACTL register; the host sends data in blocks of size BLKSZ each. Program SDIO_CMDAGMT register with the data address, where data should be written. Program the SDIO command control register (SDIO_CMDCTL): CMDIDX with 24, CMDRESP with 1 (SDIO card host waits for a short response); CSMEN with '1' (enable to send a command). Other fields are their reset value.

When the CMDRECV flag is set, program the SDIO data control register (SDIO_DATACTL): DATAEN with 1 (enable to send data); DATADIR with 0 (from controller to card); TRANSMOD with 0 (block data transfer); DMAEN with 1 (DMA enabled); BLKSZ with 0x9 (512 bytes). Other bits don't care.

Wait for DTBLKEND flag is set. Check that no channels are still enabled by polling the DMA Interrupt Flag register.

It consists the following subunits:

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. The transmit FIFO is used when write data to card and TXRUN in SDIO_STAT register is 1. The data to be transferred is written to transmit FIFO by APB2 bus, the data unit in SDIO adapter read data from transmit FIFO, and then send the data to card. The receive FIFO is used when read data from card and RXRUN in SDIO_STAT register is 1. The data to be transferred is read from the card and then write to receive FIFO. The data in receive FIFO is read to APB2 bus when needed. This unit also generates FIFO flags in SDIO_STAT registers.

29.5. Card functional description

29.5.1. 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 (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 MMC and SD card. The host can use CMD1 (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 MMC4.2 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 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(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.

29.5.2. 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 29-7. Command Token Format***, needing a transmission time of 1.92μs (25 MHz) 0.96μs(50 MHz) and 0.92us(52 MHz).

Figure 29-7. Command Token Format

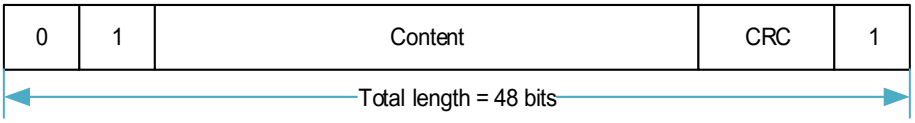


Table 29-2. Command format

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 29-3. Card command classes (CCCs)***). Each class supports a set of card functionalities. ***Table 29-3. 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 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.

For CE-ATA device, the device shall support the MMC commands required to achieve the transfer state during device initialization. Other interface configuration settings, such as bus width, may require additional MMC commands also be supported. See the MMC reference. CE-ATA makes use of the following MMC commands: CMD0 - GO_IDLE_STATE, CMD12 - STOP_TRANSMISSION, CMD39 - FAST_IO, CMD60 - RW_MULTIPLE_REGISTER, CMD61 - RW_MULTIPLE_BLOCK. GO_IDLE_STATE (CMD0), STOP_TRANSMISSION (CMD12), and FAST_IO (CMD39) are as defined in the MMC reference. RW_MULTIPLE_REGISTER (CMD60) and RW_MULTIPLE_BLOCK (CMD61) are MMC commands defined by CE-ATA.

Table 29-3. 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**	**switch**	**reserved**
CMD0	M	+											
CMD1	M	+											
CMD2	M	+											
CMD3	M	+											
CMD4	M	+											
CMD5	O										+		
CMD6	M											+	
CMD7	M	+											

CMD8	M	+											
CMD9	M	+											
CMD10	M	+											
CMD11	M		+										
CMD12	M	+											
CMD13	M	+											
CMD14	M	+											
CMD15	M	+											
CMD16	M			+		+			+				
CMD17	M			+									
CMD18	M			+									
CMD19	M	+											
CMD20	M				+								
CMD23	M			+		+							
CMD24	M					+							
CMD25	M					+							
CMD26	M					+							
CMD27	M					+							
CMD28	M							+					
CMD29	M							+					
CMD30	M							+					
CMD32	M						+						
CMD33	M						+						
CMD34	O											+	
CMD35	O											+	
CMD36	O											+	
CMD37	O											+	
CMD38	M						+						
CMD39											+		
CMD40											+		
CMD42									+				
CMD50	O											+	
CMD52	O										+		
CMD53	O										+		
CMD55	M									+			
CMD56	M									+			
CMD57	O											+	
CMD60	M									+			
CMD61	M									+			
ACMD6	M									+			
ACMD13	M									+			
ACMD22	M									+			
ACMD23	M									+			

ACMD41	M									+			
ACMD42	M									+			
ACMD51	M									+			

Note: 1.CMD1, CMD11, CMD14, CMD19, CMD20, CMD23, CMD26, CMD39 and CMD40 are only available for MMC.CMD5, CMD32-34, CMD50, CMD52, CMD53, CMD57 and ACMDx are only available for SD card. CMD60, CMD61 are only available for CE-ATA device.

2. All the ACMDs shall be preceded with APP_CMD command (CMD55).

3. CMD8 has different meaning for MMC and SD memory.

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 29-4. Basic commands (class 0)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all cards to idle state
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)
CMD4	bc	[31:16] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards
CMD5	bcr	[31:25]reserved bits [24]S18R [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.
CMD6	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Only for 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 [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 [11:8]supply voltage(VHS) [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 MMC only. The card sends its EXT_CSD register as a block of data.
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card identification (CID) on CMD the line.
CMD12	ac	[31:0] stuff bits	R1b	STOP TRANSMISSION	Forces the card to stop transmission
CMD13	ac	[31:16] RCA [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 [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 29-5. Block-Oriented read commands (class 2)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block	R1	SET_BLOCKLEN	In the case of a Standard

Cmd index	type	argument	Response format	Abbreviation	Description
		length			Capacity SD and MMC, this command sets the block length (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. 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 MMC, this command reads a block of the size selected by the SET_BLOCKLEN command. 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.
Note: The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register					

Table 29-6. Stream read commands (class 1) and stream write commands (class 3)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD11	adtc	[31:0] data	R1	READ_DAT_UNTI	Reads data stream from the card,

Cmd index	type	argument	Response format	Abbreviation	Description
		address		L_STOP	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 29-7. 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 29-5. 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.
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). 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 29-8. 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.(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.(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 MMC. 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 29-9. 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

Cmd index	type	argument	Response format	Abbreviation	Description
					write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE). 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 29-10. 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 29-5. 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. Reserved bits in the argument and in Lock Card Data Structure shall be set to 0.

Table 29-11. Application-specific commands (class 8)

Cmd index	type	argument	Response format	Abbreviation	Description
ACMD41	bcr	[31]reserved bit [30]HCS [29:24]reserved bits [23:0]V_{DD} Voltage Window(OCR[23:	R3	SD_SEND_OP_COND	Sends host capacity support information (HCS) and asks the accessed card to send its operating condition register(OCR) content in the response. HCS is effective

Cmd index	type	argument	Response format	Abbreviation	Description
		0])			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.
CMD60	adtc	[31] WR [23:18] Address [7:2] Byte Count Other bits are reserved bits.	R1(read)/ R1b(write)	RW_MULTIPLE_REGISTER	Read or write register in address range.
CMD61	adtc	[31] WR [15:0] Data Unit Count Other bits are reserved bits	R1(read)/ R1b(write)	RW_MULTIPLE_BLOCK	Read or write data block in address range.
Note: 1.ACMDx is Application-specific Commands for SD memory. 2. CMD60, CMD61 are Application-specific Commands for CE-ATA device.					

Table 29-12. 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	R4	FAST_IO	Used to write and read 8 bit (register) data fields. The command addresses a card and a register and provides the

Cmd index	type	argument	Response format	Abbreviation	Description
		address [7:0] register data			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 not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode
CMD52	adtc	[31] R/W Flag [30:28] Function Number [27] RAW Flag [26] Stuff Bits [25:9] Register Address [8] Stuff Bits [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 [30:28] Function Number [27] Block Mode [26] OP code [25:9] Register Address [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 MMC. 2. CMD52, CMD53 are only for SD I/O card.					

Table 29-13. Switch function commands (class 10)

Cmd index	type	argument	Response format	Abbreviation	Description
CMD6	adtc	[31] Mode 0:Check function 1:Switch function [30:24] reserved [23:20] reserved for function group 6 (0h or Fh) [19:16] reserved for function group 5 (0h or Fh) [15:12] reserved for function group 4 (0h or Fh) [11:8] reserved for function group 3 (0h or Fh) [7:4] function group 2 for command system [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).

29.5.3. 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.

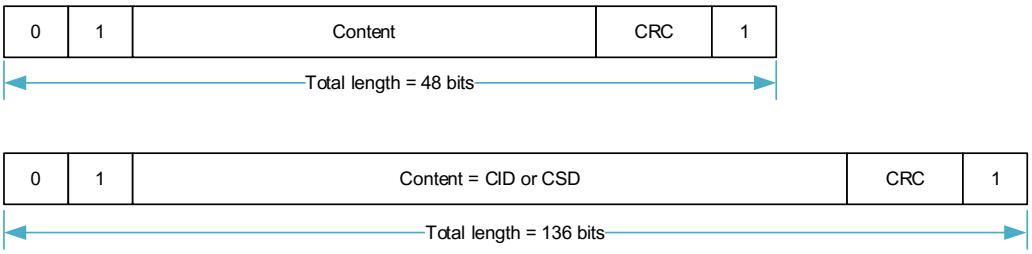
The SD Memory Card support five types of them, R1 / R1b, R2, R3, R6, R7. And the SD I/O Card and MMC supports additional response types named R4 and R5, but they are not exactly the same for SD I/O Card and MMC.

Responses format

Responses have two formats, as show in *Figure 29-8. 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 29-8. 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 29-14. 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 29-15. 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 (MMC). The response of different cards may have a little different.

Table 29-16. 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 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 29-17. Response R4 for 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 29-18. 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'	'111111'	x	x	x	'000'	x	x	'1111111'	1
descripti	start	transmiss	Reser	C	Number	Memory	Stuff	S18	I/O	Reser	end

on	bit	ion bit	ved		of I/O functions	Present	Bits	A	OCR	ved	bit

R5 (Interrupt request)

For 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 29-19. Response R5 for 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 29-20. 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 29-21. 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 bit	transmission bit	CMD3	New published RCA of the card	card status bits:23,22,19,12:0	CRC7	end 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 29-22. 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

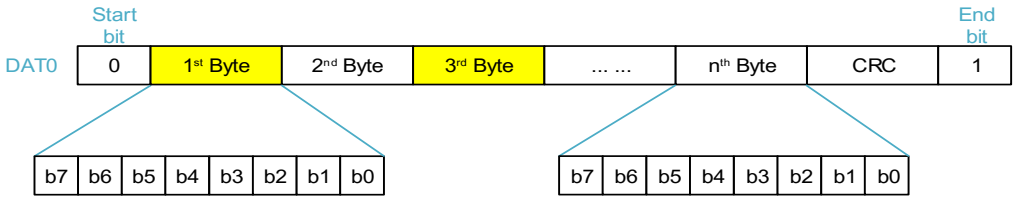
29.5.4. Data packets format

There are 3 data bus mode, 1-bit, 4-bit and 8-bit width. 1-bit mode is mandatory, 4-bit and 8-bit mode is optional. Although using 1-bit mode, DAT3 also need to notify card current working mode is SDIO or SPI, when card reset and initialize.

1-bit data packet format

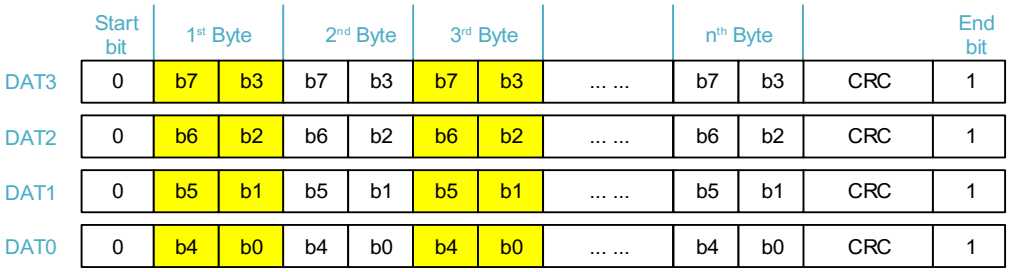
After card reset and initialize, only DAT0 pin is used to transfer data. And other pin can be used freely. *Figure 29-9. 1-bit data bus width*, *Figure 29-10. 4-bit data bus width* and *Figure 29-11. 8-bit data bus width* show the data packet format when data bus wide is 1-bit, 4-bit and 8-bit.

Figure 29-9. 1-bit data bus width



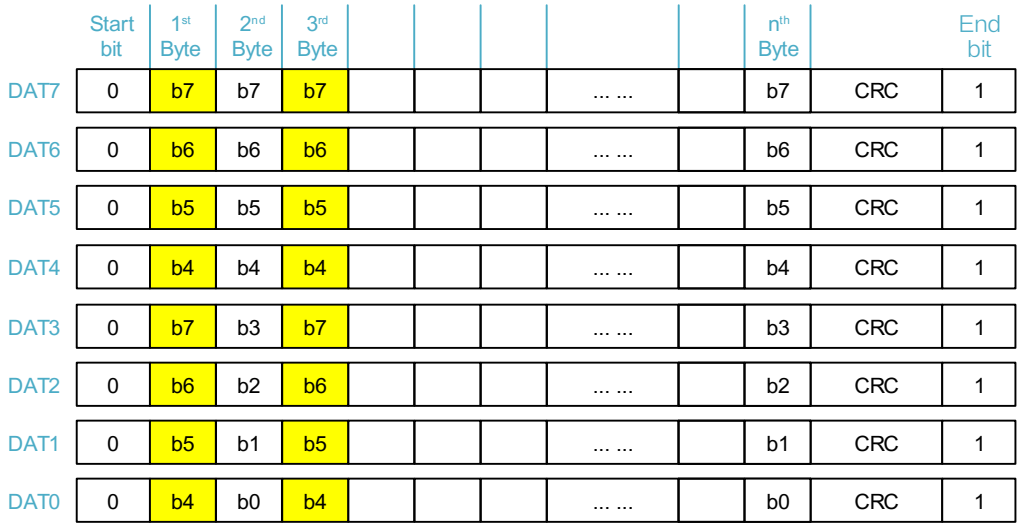
4-bit data packet format

Figure 29-10. 4-bit data bus width



8-bit data packet format

Figure 29-11. 8-bit data bus width



29.5.5. 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 29-23. Card status

Bits	Identifier	Type	Value	Description	Clear Condition
31	OUT_OF_RANGE	ERX	'0'= no error '1'= error	The command's argument was out of the allowed range for this card.	C
30	ADDRESS_ERROR	ERX	'0'= no error '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 '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 '1'= error	An error in the sequence of erase commands occurred.	C
27	ERASE_PARAM	ERX	'0'= no error '1'= error	An invalid selection of write-blocks for erase occurred.	C
26	WP_VIOLATION	ERX	'0'= not protected '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 '1' = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_FAILED	ERX	'0'= no error '1'= error	Set when a sequence or password error has been detected in lock/unlock card command.	C
23	COM_CRC_ERROR	ER	'0'= no error '1'= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	ER	'0'= no error '1'= error	Command not legal for the card state.	B
21	CARD_ECC_FAILED	ERX	'0'= success '1'= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	ERX	'0'= no error '1'= error	Internal card controller error.	C
19	ERROR	ERX	'0'= no error	A general or an unknown error	C

Bits	Identifier	Type	Value	Description	Clear Condition
			'1'= error	occurred during the operation.	
18	UNDERRUN	ERX	'0'= no error '1'= error	Only for MMC. The card could not sustain data transfer in stream read mode.	C
17	OVERRUN	ERX	'0'= no error '1'= error	Only for MMC. The card could not sustain data programming in stream write mode.	C
16	CID/ CSD_OVERWRITE	ERX	'0'= no error '1'= error	Can be either one of the following errors: - The read only section of the CSD does not match the card content. - 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 '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 '1'= disabled	The command has been executed without using the internal ECC.	A
13	ERASE_RESET	SR	'0'= cleared '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 1 = ready 2 = identification 3 = stand by 4 = transfer 5 = send data 6 = receive data 7 = programming 8 = disconnect 9-14 = reserved 15 = reserved for I/O mode	The state of the card when receiving the command. If the 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.	B

Bits	Identifier	Type	Value	Description	Clear Condition
8	READY_FOR_DATA	SX	'0'= not ready '1'= ready	Corresponds to buffer empty signaling on the bus.	A
7	SWITCH_ERROR	EX	'0'= no error '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 '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 '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 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 29-24. SD status

Bits	Identifier	Type	Value	Description	Clear Condition
[511:510]	DAT_BUS_WIDTH	SR	'00'= 1 (default) '01'= reserved '10'= 4 bit width '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 '1'= In Secured Mode	Card is in Secured Mode of operation (refer to the "SD Security Specification").	A
[508:	reserved				

Bits	Identifier	Type	Value	Description	Clear Condition
496]					
[495:480]	SD_CARD_TYPE	SR	The following cards are currently defined: '0000'= Regular SD RD/WR Card. '0001'= SD ROM Card '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
[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 29-25. Performance move field*.

Table 29-25. 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 29-26. 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 29-26. AU_SIZE field***. The card can set any AU size specified in ***Table 29-27. Maximum AU size*** that is less than or equal to the maximum AU size. The card should set smaller AU size as possible.

Table 29-27. 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 29-28. 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.

$$\text{Erase timeout of X AU} = \frac{T_{ERASE}}{N_{ERASE}} * X + T_{OFFSET} \quad (28\text{-}1)$$

Table 29-29. 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 29-30. Erase offset field

ERASE_OFFSET	Value Definition
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

29.6. Programming sequence

29.6.1. 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 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 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 MMC, CE-ATA, SD, or SD I/O. All types of SD I/O cards are supported, they are, 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, MMC(CE-ATA), 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 MMC or CE-ATA.

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.

- MMC/CE-ATA - Send CMD0, CMD1, CMD2, CMD3.

4. Identify the MMC/CE-ATA device.

- CPU should query the byte 504 (S_CMD_SET) of EXT_CSD register by sending CMD8. If bit 4 is set to 1, then the device supports ATA mode.

- If ATA mode is supported, the CPU should select the ATA mode by setting the ATA bit (bit 4) in the EXT_CSD register slice 191(CMD_SET) to activate the ATA command set. The CPU selects the command set using the SWITCH (CMD6) command.

- In the presence of a CE-ATA device, the FAST_IO (CMD39) and RW_MULTIPLE_REGISTER (CMD60) commands will succeed and the returned data will be the CE-ATA reset signature.

29.6.2. 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.

29.6.3. 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 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. For CE-ATA, first use CMD60 to write the ATA task file, then use CMD61 to write the data. After writing to the CMD register, host starts executing a command, when the command is sent to the bus, the CMDRECV flag is set.

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.

29.6.4. 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 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. For CE-ATA, first using CMD60 to write the ATA task file, then using CMD 61 to read the data. 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.

29.6.5. Stream write and stream read (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:

$$\text{max write frequence} = \min\left(\text{TRAN_SPEED}, \frac{8*2^{\text{WRITE_BL_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W_FACTOR}}\right) \quad (28\text{-}2)$$

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:

$$\text{max read frequence} = \min\left(\text{TRAN_SPEED}, \frac{8*2^{\text{READ_BL_LEN}}-100*\text{NSAC}}{\text{TAAC}*\text{R2W_FACTOR}}\right) \quad (28\text{-}3)$$

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

29.6.6. Erase

The erasable unit of the 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.

29.6.7. Bus width selection

After the host has verified the functional pins on the bus it should change the bus width configuration.

For 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).

29.6.8. 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***.

29.6.9. 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 29-31. Lock card data structure*** describes the structure of the command data block.

Table 29-31. 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.

29.7. Specific operations

29.7.1. SD I/O specific operations

The SD I/O only card and SD I/O combo card support these specific operations:

Read Wait operation

Suspend/resume operation

Interrupts

The SD I/O supports these operations only if the SDIO_DATACTL[11] bit is set, except for read suspend that does not need specific hardware implementation.

SD I/O read wait operation

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation allows a host to signal a card that is executing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O card. To determine if a card supports the Read Wait protocol, the host shall test SRW capability bit in the Card Capability byte of the CCCR. The timing for Read Wait is based on the Interrupt Period. If a card does not support the Read Wait protocol, the only means a host has to stall (not abort) data in the middle of a read multiple command is to control the SDIO_CLK. The limitation of this method is that with the clock stopped, the host cannot issue any commands, and so cannot perform other operations during the delay time. Read Wait support is mandatory for the card to support suspend/resume. *Figure 29-12. Read wait control by stopping SDIO_CLK* and *Figure 29-13. Read wait operation using*

SDIO_DAT[2] show the Read Wait mode about stop the SDIO_CLK and use SDIO_DAT[2].

Figure 29-12. Read wait control by stopping SDIO_CLK

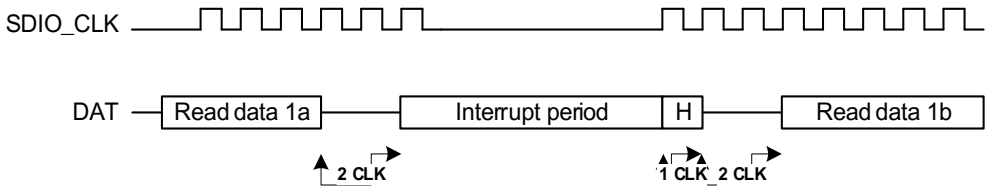
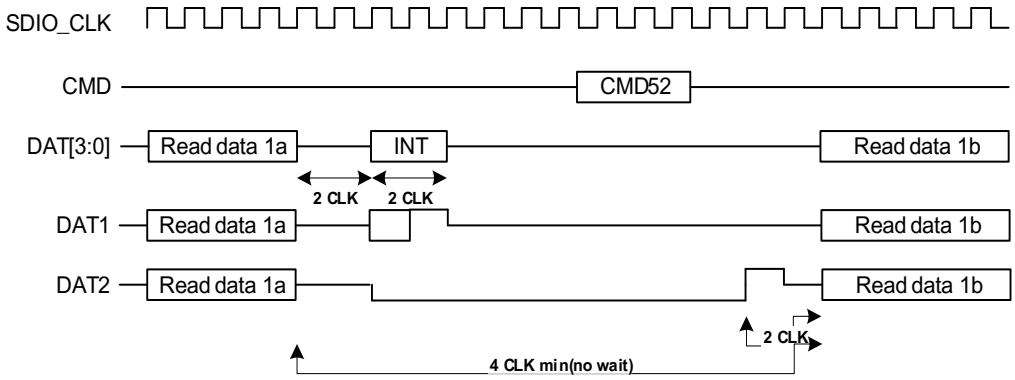


Figure 29-13. Read wait operation using SDIO_DAT[2]



We can start the Read Wait interval before the data block is received: when the data unit is enabled (SDIO_DATACTL[0] bit set), the SD I/O specific operation is enabled (SDIO_DATACTL[11] bit set), Read Wait starts (SDIO_DATACTL[10] = 0 and SDIO_DATACTL[8] = 1) and data direction is from card to SD I/O (SDIO_DATACTL[1] = 1), the DSM directly moves from Idle to Read Wait. In Read Wait the DSM drives SDIO_DAT[2] to 0 after 2 SDIO_CLK clock cycles. In this state, when you set the RWSTOP bit (SDIO_DATACTL[9]), the DSM remains in Wait for two more SDIO_CLK clock cycles to drive SDIO_DAT[2] to 1 for one clock cycle. The DSM then starts waiting again until it receives data from the card. The DSM will not start a Read Wait interval while receiving a block even if Read Wait start is set: the Read Wait interval will start after the CRC is received. The RWSTOP bit has to be cleared to start a new Read Wait operation. During the Read Wait interval, the SDIO can detect SD I/O interrupts on SDIO_DAT[1].

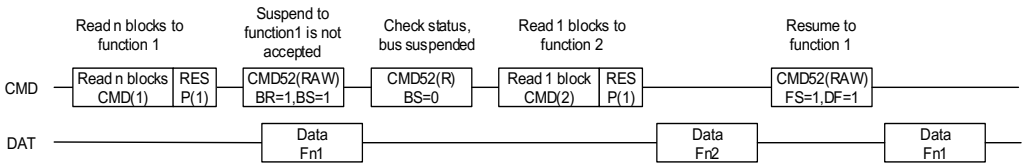
SD I/O suspend/resume operation

Within a multi-function SD I/O or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SD I/O and combo cards can implement the optional concept of suspend/resume. If a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function or memory. Once this higher-priority transfer is completed, the original transfer is re-started where it left off (resume).

Figure 29-14. Function2 read cycle inserted during function1 multiple read cycle shows a condition where the first suspend request is not immediately accepted. The host then checks the status of the request with a read and determines that the bus has now been

released (BS=0). At this time, a read to function 2 is started. Once that single block read is completed, the resume is issued to function, causing the data transfer to resume (DF=1).

Figure 29-14. Function2 read cycle inserted during function1 multiple read cycle



When the host sends data to the card, the host can suspend the write operation. The SDIO_CMDCTL[11] bit is set and indicates to the CSM that the current command is a suspend command. The CSM analyzes the response and when the response is received from the card (suspend accepted), it acknowledges the DSM that goes Idle after receiving the CRC token of the current block.

To suspend a read operation, the DSM waits in the WaitR state, when the function to be suspended sends a complete packet just before stopping the data transaction. The application should continue reading receive FIFO until the FIFO is empty, and the DSM goes Idle state automatically.

Interrupts

In order to allow the SD I/O card to interrupt the host, an interrupt function is added to a pin on the SD interface. Pin number 8, which is used as SDIO_DAT[1] when operating in the 4-bit SD mode, is used to signal the card's interrupt to the host. The use of interrupt is optional for each card or function within a card. The SD I/O interrupt is "level sensitive", that is, the interrupt line shall be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period. Once the host has serviced the interrupt, it is cleared via function unique I/O operation.

When setting the SDIO_DATACTL[11] bit SD I/O interrupts can detect on the SDIO_DAT[1] line.

Figure 29-15. Read Interrupt cycle timing shows the timing of the interrupt period for single data transaction read cycles.

Figure 29-15. Read Interrupt cycle timing

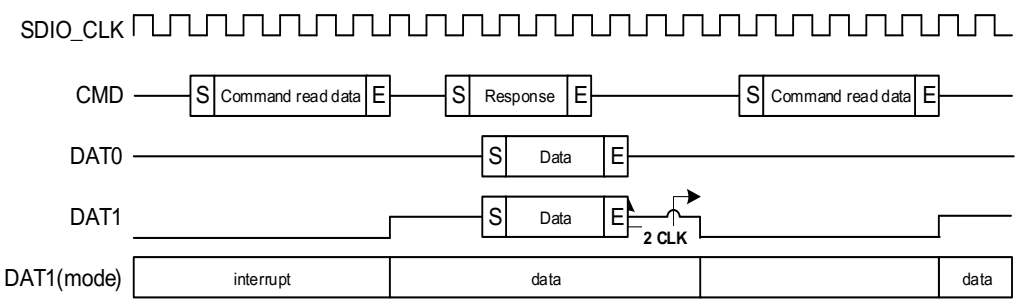
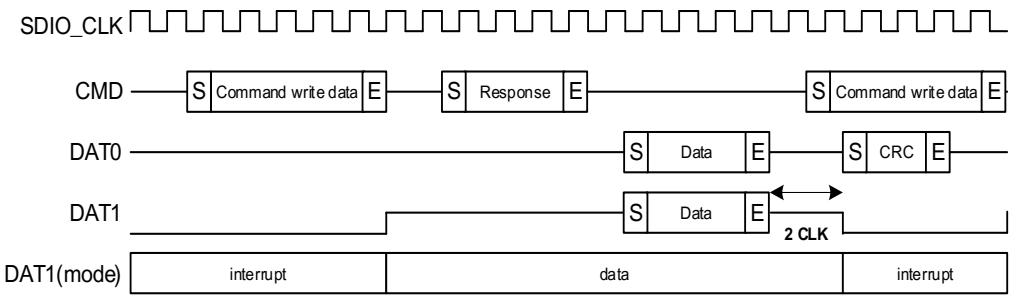


Figure 29-16. Write interrupt cycle timing



When transferring multiple blocks of data in the 4-bit SD mode, a special definition of the interrupt period is required. In order to allow the highest speed of communication, the interrupt period is limited to a 2-clock interrupt period. Card that wants to send an interrupt signal to the host shall assert DAT1 low for the first clock and high for the second clock. The card shall then release DAT1 into the hi-Z State. ***Figure 29-17. Multiple block 4-Bit read interrupt cycle timing*** shows the operation for an interrupt during a 4-bit multi-block read and ***Figure 29-18. Multiple block 4-Bit write interrupt cycle timing*** shows the operation for an interrupt during a 4-bit multi-block write

Figure 29-17. Multiple block 4-Bit read interrupt cycle timing

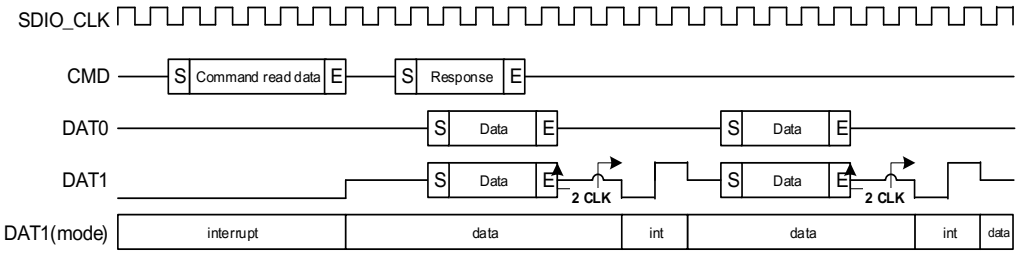
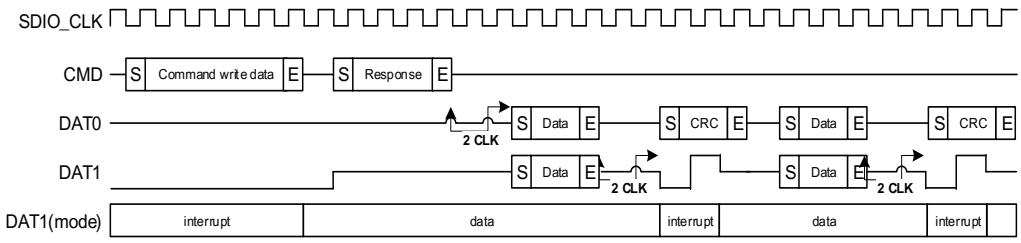


Figure 29-18. Multiple block 4-Bit write interrupt cycle timing



29.7.2. CE-ATA specific operations

The CE-ATA device supports these specific operations:

Receive command completion signal

Send command completion disable signal

The SDIO supports these operations only when SDIO_CMDCTL[14] is set.

Command completion signal

CE-ATA defines a command completion signal that the device uses to notify the host upon normal ATA command completion or when ATA command termination has occurred due to an error condition the device has encountered.

If the 'enable CMD completion' bit SDIO_CMDCTL[12] is set and the 'not interrupt Enable' bit SDIO_CMDCTL[13] is reset, the CSM waits for the command completion signal in the Waitcompl state.

When start bit is received on the CMD line, the CSM enters the Idle state. No new command can be sent for 7 bit cycles. Then, for the last 5 cycles (out of the 7) the CMD line is driven to'1' in push-pull mode.

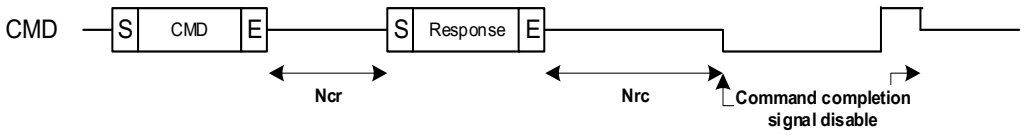
After the host detects a command completion signal from the device, it should issue a FAST_IO (CMD39) command to read the ATA Status register to determine the ending status for the ATA command.

Command completion disable signal

The host may cancel the ability for the device to return a command completion signal by issuing the command completion signal disable. The host shall only issue the command completion signal disable when it has received an R1b response for an outstanding RW_MULTIPLE_BLOCK (CMD61) command.

Command completion signal disable is sent 8 bit cycles after the reception of a short response if the 'enable CMD completion' bit, SDIO_CMDCTL[12] is not set and the 'not interrupt Enable' bit SDIO_CMDCTL[13] is reset.

Figure 29-19. The operation for command completion disable signal



29.8. Register definition

SDIO base address: 0x4001 2C00

29.8.1. Power control register (SDIO_PWRCTL)

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														PWRCTL[1:0]	
														rw	

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1:0	PWRCTL[1:0]	SDIO power control bits. These bits control the SDIO state, card input or output. 00: SDIO power off: SDIO cmd/data state machine reset to IDLE, clock to card stopped, no cmd/data output to card 01: Reserved 10: Reserved 11: SDIO Power on

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

29.8.2. 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
DIV[8]	Reserved														
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HWCLKEN	CLKEDGE	BUSMODE[1:0]		CLKBYP	CLKPWRSAV	CLKEN	DIV[7:0]							
	rw	rw	rw		rw	rw	rw	rw							

Bits	Fields	Descriptions
31	DIV[8]	MSB of Clock division This field defines the MSB division between the input clock (SDIOCLK) and the output clock, refer to bit 7:0 of SDIO_CLKCTL
30:15	Reserved	Must be kept at reset value.
14	HWCLKEN	Hardware Clock Control enable bit If this bit is set, hardware controls the SDIO_CLK on/off depending on the system bus is very busy or not. There is no underrun/overrun error when this bit is set, because hardware can close the SDIO_CLK when almost underrun/overrun. 0: HW Clock control is disabled 1: HW Clock control is enabled
13	CLKEDGE	SDIO_CLK clock edge selection bit 0: Select the rising edge of the SDIOCLK to generate SDIO_CLK 1: Select the falling edge of the SDIOCLK to generate SDIO_CLK
12:11	BUSMODE[1:0]	SDIO card bus mode control bit 00: 1-bit SDIO card bus mode selected 01: 4-bit SDIO card bus mode selected 10: 8-bit SDIO card bus mode selected
10	CLKBYP	Clock bypass enable bit This bit defines the SDIO_CLK is directly SDIOCLK or not. 0: NO bypass, the SDIO_CLK refers to DIV bits in SDIO_CLKCTL register. 1: Clock bypass, the SDIO_CLK is directly from SDIOCLK (SDIOCLK/1).
9	CLKPWRSAV	SDIO_CLK clock dynamic switch on/off for power saving. This bit controls SDIO_CLK clock dynamic switch on/off when the bus is idle for power saving 0: SDIO_CLK clock is always on 1: SDIO_CLK closed when bus idle
8	CLKEN	SDIO_CLK clock output enable bit 0: SDIO_CLK is disabled 1: SDIO_CLK is enabled
7:0	DIV[7:0]	Clock division This field and DIV[8] bit defines the division factor to generator SDIO_CLK clock to card. The SDIO_CLK is divider from SDIOCLK if CLKBYP bit is 0, and the SDIO_CLK frequency = SDIOCLK / (DIV[8:0] + 2).

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

29.8.3. Command argument register (SDIO_CMDAGMT)

Address offset: 0x08

Reset value: 0x0000 0000

This register defines 32 bit 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 This field defines the SDIO 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 writting SDIO_CMDCTL register when sending a command.

29.8.4. 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															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ATAEN	NINTEN	ENCMDC	SUSPEND	CSMEN	WAITDEND	INTWAIT	CMDRESP[1:0]		CMDIDX[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw		rw					

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	ATAEN	CE-ATA command enable(CE-ATA only) If this bit is set, the host enters the CE-ATA mode, and the CSM transfers CMD61. 0: CE-ATA disable 1: CE-ATA enable
13	NINTEN	No CE-ATA Interrupt (CE-ATA only) This bit defines if there is CE-ATA interrupt or not. This bit is only used when CE-ATA card. 0: CE-ATA interrupt enable

		1: CE_ATA interrupt disable
12	ENCMDC	CMD completion signal enabled (CE-ATA only) This bit defines if there is command completion signal or not in CE-ATA card. 0: no completion signal 1: have completion signal
11	SUSPEND	SD I/O suspend command(SD I/O only) This bit defines whether the CSM to send a suspend command or not. This bit is only used for SDIO card. 0: no effect 1: suspend command
10	CSMEN	Command state machine (CSM) enable bit 0: Command state machine disable (stay on CS_Idle) 1: Command state machine enable
9	WAITDEND	Waits for ends of data transfer. If this bit is set, the command state machine starts to send a command must wait the end of data transfer. 0: no effect 1: Wait the end of data transfer
8	INTWAIT	Interrupt wait instead of timeout This bit defines the command state machine to wait card interrupt at CS_Wait state in command state machine. If this bit is set, no command wait timeout generated. 0: Not wait interrupt. 1: Wait interrupt.
7:6	CMDRESP[1:0]	Command response type bits These bits define the response type after sending a command message. 00: No response 01: Short response 10: No response 11: Long response
5:0	CMDIDX[5:0]	Command index This field defines the command index to be sent to SDIO card.

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

29.8.5. Command index response register (SDIO_RSPCMDIDX)

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										RSPCMDIDX[5:0]					
										r					

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.

29.8.6. Response register (SDIO_RESPx) (x=0...3)

Address offset: 0x14 + 4 * x

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 29-32. 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 29-32. SDIO_RESPx register at different response type

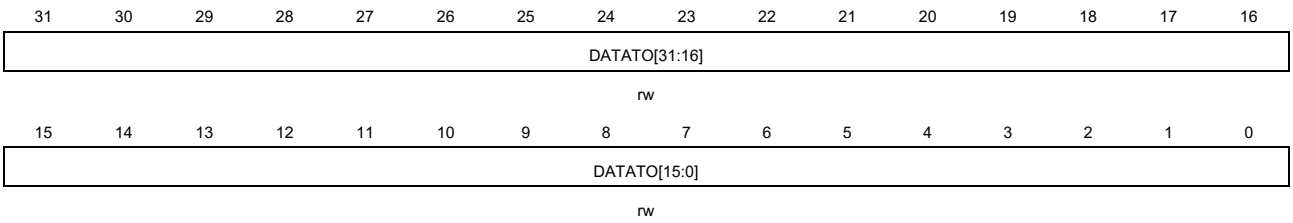
Register	Short response	Long response
SDIO_RESP0	Card response[31:0]	Card response[127:96]
SDIO_RESP1	reserved	Card response [95:64]
SDIO_RESP2	reserved	Card response [63:32]
SDIO_RESP3	reserved	Card response [31:1],plus bit 0

29.8.7. Data timeout register (SDIO_DATATO)

Address offset: 0x24

Reset value: 0x0000 0000

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



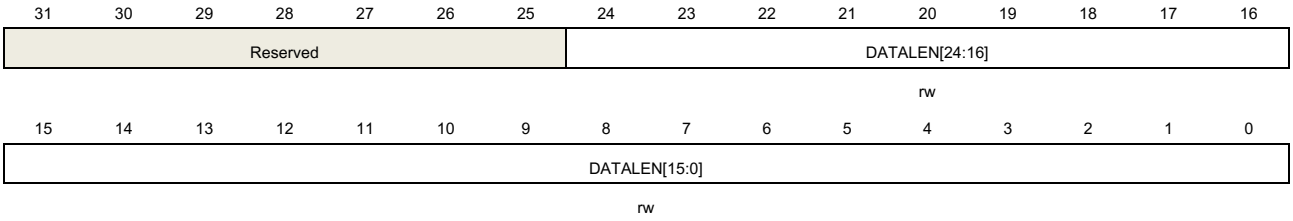
Bits	Fields	Descriptions
31:0	DATATO[31:0]	Data timeout period These bits define the data timeout period count by SDIO_CLK. When the DSM enter the state WaitR or BUSY, the internal counter which loads from this register starts decrement. The DSM timeout and enter the state Idle and set the DTTMOUT flag when the counter decreases to 0.

Note: The data timer register and the data length register must be updated before being written to the data control register when need a data transfer.

29.8.8. **Data length** register (SDIO_DATALEN)

Address offset: 0x28
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.
24:0	DATALEN[24:0]	Data transfer length This register defined the number of bytes to be transferred. When the data transfer starts, the data counter loads this register and starts decrement.

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 timer register and the data length register must be updated before being written to the data control register when need a data transfer.

29.8.9. **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				IOEN	RWTYPE	RWSTOP	RWEN	BLKSZ[3:0]				DMAEN	TRANSMOD	DATADIR	DATAEN
				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	IOEN	SD I/O specific function enable(SD I/O only) 0: Not SD I/O specific function 1: SD I/O specific function
10	RWTYPE	Read wait type(SD I/O only) 0: Read Wait control using SDIO_DAT[2] 1: Read Wait control by stopping SDIO_CLK
9	RWSTOP	Read wait stop(SD I/O only) 0: No effect 1: Stop the read wait process if RWEN bit is set
8	RWEN	Read wait mode enabled(SD I/O only) 0: Read wait mode is disabled 1: Read wait mode is enabled
7:4	BLKSZ[3:0]	Data block size These bits defined the block size when data transfer is block transfer. 0000: block size = 2^0 = 1 byte 0001: block size = 2^1 = 2 bytes 0010: block size = 2^2 = 4 bytes 0011: block size = 2^3 = 8 bytes 0100: block size = 2^4 = 16 bytes 0101: block size = 2^5 = 32 bytes 0110: block size = 2^6 = 64 bytes 0111: block size = 2^7 = 128 bytes 1000: block size = 2^8 = 256 bytes 1001: block size = 2^9 = 512 bytes 1010: block size = 2^{10} = 1024 bytes 1011: block size = 2^{11} = 2048 bytes 1100: block size = 2^{12} = 4096 bytes 1101: block size = 2^{13} = 8192 bytes 1110: block size = 2^{14} = 16384 bytes 1111: reserved
3	DMAEN	DMA enable bit

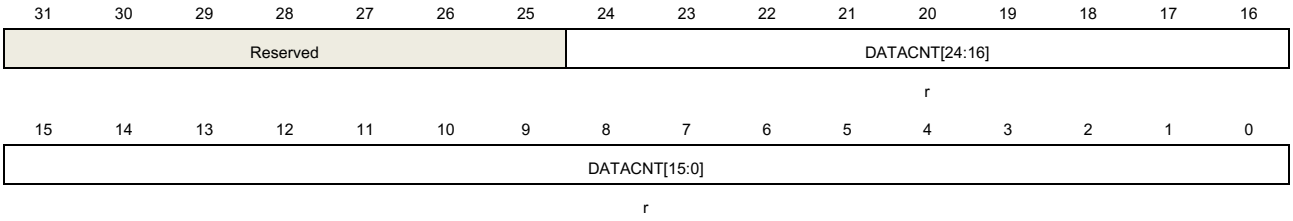
		0: DMA is disabled. 1: DMA is enabled.
2	TRANSMOD	Data transfer mode 0: Block transfer 1: Stream transfer or SDIO multibyte transfer
1	DATADIR	Data transfer direction 0: Write data to card. 1: Read data from card.
0	DATAEN	Data transfer enable bit Write 1 to this bit to start data transfer regardless this bit is 0 or 1. The DSM moves to Readwait state if RWEN is set or to the WaitS, WaitR state depend on DATADIR bit. Start a new data transfer, it not need to clear this bit to 0.

Note: Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

29.8.10. 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 Read-only bits field. When these bits are read, the number of remaining data bytes to be transferred is returned.

29.8.11. 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 [23:22, 10:0] can only be cleared by writing 1 to the corresponding bit in interrupt clear register (SDIO_INTC).

The flags of bit [21:11] 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								ATAEND	SDIOINT	RXDTVAL	TXDTVAL	RFE	TFE	RFF	TFF
								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	RXRUN	TXRUN	CMDRUN	DTBLKEND	STBITE	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
31:24	Reserved	Must be kept at reset value.
23	ATAEND	CE-ATA command completion signal received (only for CMD61)
22	SDIOINT	SD I/O interrupt received
21	RXDTVAL	Data is valid in receive FIFO
20	TXDTVAL	Data is valid in transmit FIFO
19	RFE	Receive FIFO is empty
18	TFE	Transmit FIFO is empty When HW Flow control is enabled, TFE signals becomes activated when the FIFO contains 2 words.
17	RFF	Receive FIFO is full When HW Flow control is enabled, RFF signals becomes activated 2 words before the FIFO is full.
16	TFF	Transmit FIFO is full
15	RFH	Receive FIFO is half full: at least 8 words can be read in the FIFO
14	TFH	Transmit FIFO is half empty: at least 8 words can be written into the FIFO
13	RXRUN	Data reception in progress
12	TXRUN	Data transmission in progress
11	CMDRUN	Command transmission in progress
10	DTBLKEND	Data block sent/received (CRC check passed)
9	STBITE	Start bit error in the bus.
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 The data timeout period depends on the SDIO_DATATO register.
2	CMDTMOUT	Command response timeout 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)

29.8.12. 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								ATAENDC	SDIOINTC	Reserved					
								w	w						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTBLKENDC	STBITEC	DTENDC	CMDSENDC	CMDRECVC	RXOREC	TXUREC	DTTMOUTC	CMDTMOUTC	DTCRCERRC	CCRCERRC
					w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	ATAENDC	ATAEND flag clear bit Write 1 to this bit to clear the flag.
22	SDIOINTC	SDIOINT flag clear bit Write 1 to this bit to clear the flag.
21:11	Reserved	Must be kept at reset value.
10	DTBLKENDC	DTBLKEND flag clear bit Write 1 to this bit to clear the flag.
9	STBITEC	STBITE flag clear bit Write 1 to this bit to clear the flag.
8	DTENDC	DTEND flag clear bit

		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 Write 1 to this bit to clear the flag.
5	RXOREC	RXORE flag clear bit Write 1 to this bit to clear the flag.
4	TXUREC	TXURE flag clear bit Write 1 to this bit to clear the flag.
3	DTTMOUTC	DTTMOUT flag clear bit Write 1 to this bit to clear the flag.
2	CMDTMOUTC	CMDTMOUT flag clear bit Write 1 to this bit to clear the flag.
1	DTCRCERRC	DTCRCERR flag clear bit Write 1 to this bit to clear the flag.
0	CCRCERRC	CCRCERR flag clear bit Write 1 to this bit to clear the flag.

29.8.13. 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								ATAENDIE	SDIOINTIE	RXDTVALIE	TXDTVALIE	RFEIE	TFEIE	RFFIE	TFFIE
								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	RXRUNIE	TXRUNIE	CMDRUNIE	DTBLKENDIE	STBITEIE	DTENDIE	CMDSENDIE	CMDRECVIE	RXOREIE	TXUREIE	DTTMOUTIE	CMDTMOUTIE	DTCRCERRIE	CCRCERRIE
rw	rw	rw	rw	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	ATAENDIE	CE-ATA command completion signal received interrupt enable Write 1 to this bit to enable the interrupt.
22	SDIOINTIE	SD I/O interrupt received interrupt enable Write 1 to this bit to enable the interrupt.

21	RXDTVALIE	Data valid in receive FIFO interrupt enable Write 1 to this bit to enable the interrupt.
20	TXDTVALIE	Data valid in transmit FIFO interrupt enable Write 1 to this bit to enable the interrupt.
19	RFEIE	Receive FIFO empty interrupt enable Write 1 to this bit to enable the interrupt.
18	TFEIE	Transmit FIFO empty interrupt enable Write 1 to this bit to enable the interrupt.
17	RFFIE	Receive FIFO full interrupt enable Write 1 to this bit to enable the interrupt.
16	TFFIE	Transmit FIFO full interrupt enable Write 1 to this bit to enable the interrupt.
15	RFHIE	Receive FIFO half full interrupt enable Write 1 to this bit to enable the interrupt.
14	TFHIE	Transmit FIFO half empty interrupt enable Write 1 to this bit to enable the interrupt.
13	RXRUNIE	Data reception interrupt enable Write 1 to this bit to enable the interrupt.
12	TXRUNIE	Data transmission interrupt enable Write 1 to this bit to enable the interrupt.
11	CMDRUNIE	Command transmission interrupt enable Write 1 to this bit to enable the interrupt.
10	DTBLKENDIE	Data block end interrupt enable Write 1 to this bit to enable the interrupt.
9	STBITEIE	Start bit error interrupt enable Write 1 to this bit to enable the interrupt.
8	DTENDIE	Data end interrupt enable Write 1 to this bit to enable the interrupt.
7	CMDSENDIE	Command sent interrupt enable Write 1 to this bit to enable the interrupt.
6	CMDRECVIE	Command response received interrupt enable Write 1 to this bit to enable the interrupt.
5	RXOREIE	Received FIFO overrun error interrupt enable Write 1 to this bit to enable the interrupt.
4	TXUREIE	Transmit FIFO underrun error interrupt enable

		Write 1 to this bit to enable the interrupt.
3	DTTMOUTIE	Data timeout interrupt enable 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 Write 1 to this bit to enable the interrupt.
0	CCRCERRIE	Command response CRC fail interrupt enable Write 1 to this bit to enable the interrupt.

29.8.14. FIFO counter register (SDIO_FIFOCNT)

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								FIFOCNT[23:16]							
								r							

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

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23:0	FIFOCNT[23:0]	FIFO counter. These bits define the remaining number words to be written or read from the FIFO. It loads the data length register (SDIO_DATALEN[24:2] if SDIO_DATALEN is word-aligned or SDIO_DATALEN[24:2]+1 if SDIO_DATALEN is not word-aligned) when DATAEN is set, and start count decrement when a word write to or read from the FIFO.

29.8.15. FIFO data register (SDIO_FIFO)

Address offset: 0x80

Reset value: 0x0000 0000

This register occupies 32 entries of 32-bit words, the address offset is from 0x80 to 0xFC.

This register 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 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.

30. External memory controller (EXMC)

30.1. Overview

The external memory controller EXMC, is used to access a variety of external memories. By configuring the related registers, it can automatically convert AMBA memory access protocol into a specific memory access protocol, such as SRAM, PSRAM, ROM and NOR Flash. Users can also adjust the timing parameters in the configuration registers to improve 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.

30.2. Characteristics

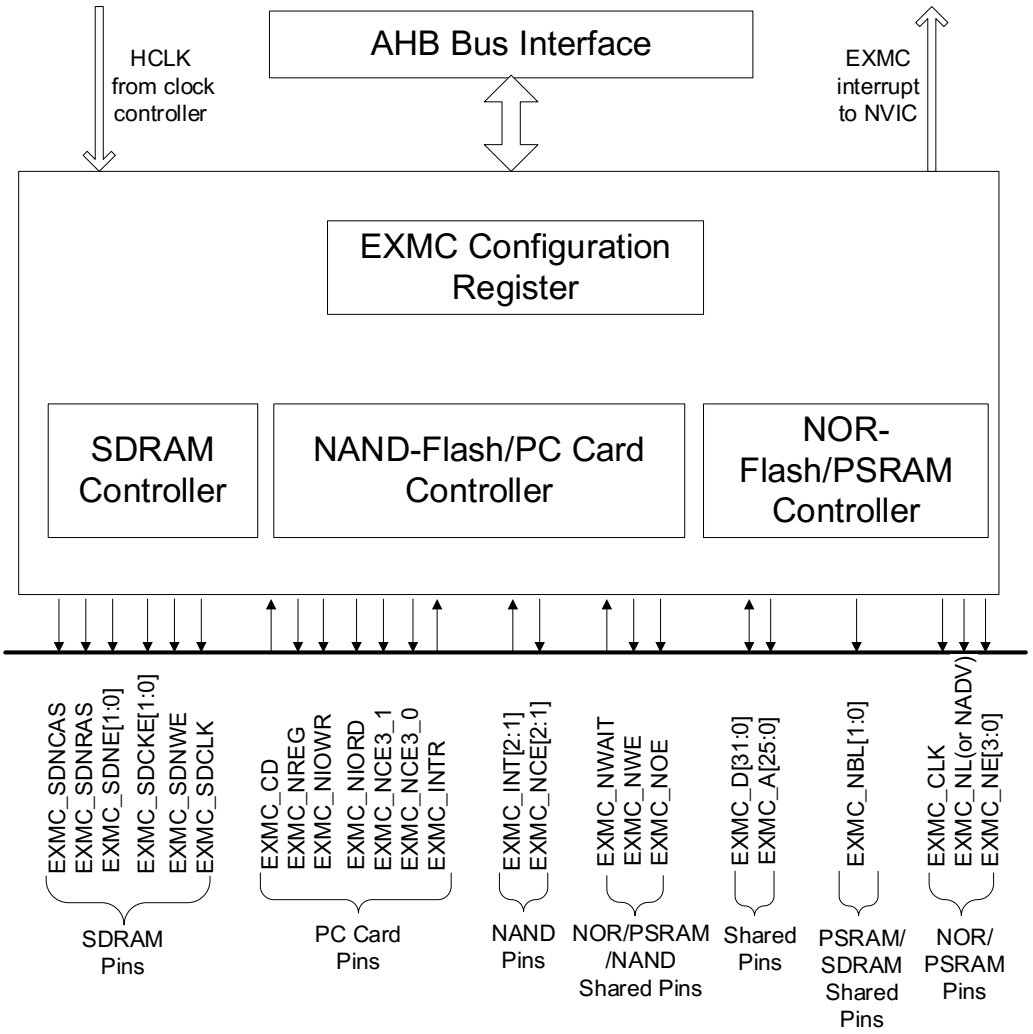
- Supported external memory:
  - SRAM
  - PSRAM/SQPI-PSRAM
  - ROM
  - NOR Flash
  - 8-bit or 16-bit NAND Flash
  - 16-bit PC Card
  - Synchronous DRAM(SDRAM)
- Protocol translation between the AMBA and the multitude of external memory protocol.
- Offering a variety of programmable timing parameters 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 if needed.
- Automatic AMBA transaction split when internal and external bus width are not compatible.

30.3. Function overview

30.3.1. Block diagram

EXMC is the combination of six modules: The AHB bus interface, EXMC configuration registers, NOR/PSRAM controller, NAND/PC Card controller, SDRAM controller and external device interface. AHB clock (HCLK) is the reference clock.

Figure 30-1. The EXMC block diagram



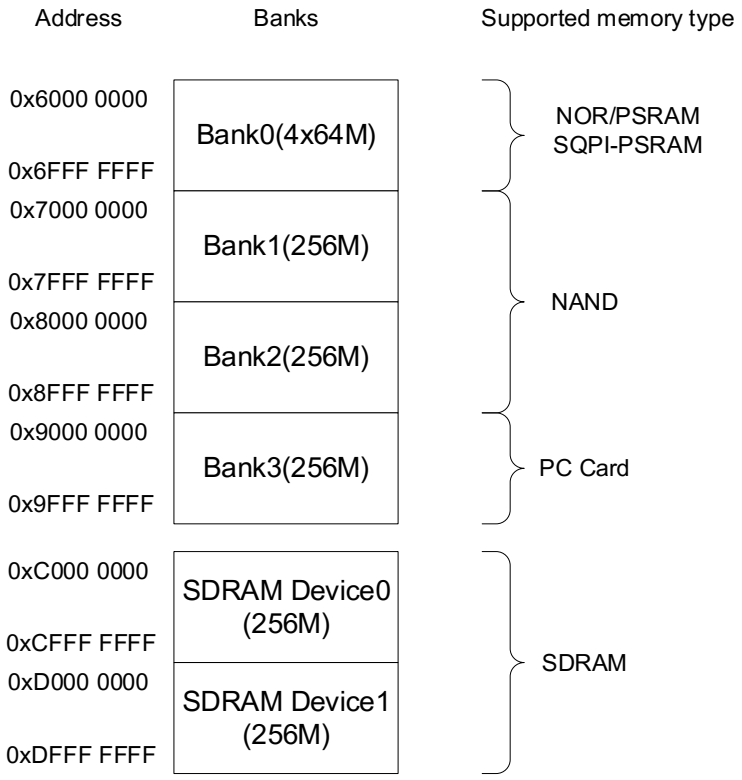
30.3.2. Basic regulation of EXMC access

EXMC is the conversion interface between AHB bus and external device protocol. 32-bit of AHB read/write accesses can be split into several consecutive 8-bit or 16-bit read/write operations respectively. In the process of data transfer, AHB access data width and memory data width may not be the same. In order to ensure consistency of data transmission, read/write accesses follow the following basic regulation.

- When the width of AHB bus equals to the memory bus width, no conversion is applied.
- When the width of AHB bus is greater than memory bus width, the AHB accesses will automatically split into several continuous memory accesses.
- When the width of AHB bus is shorter than memory bus width, if the external memory devices support the byte selection function, such as SRAM, ROM, PSRAM, the application can access the corresponding byte through EXMC_NBL [1:0]. Otherwise, write operation is prohibited, but read operation is allowed unconditionally.

30.3.3. External device address mapping

Figure 30-2. EXMC memory banks



EXMC access space is divided into multiple banks. Each bank is 256 Mbytes. The first bank (Bank0) is divided into four regions further, and each region is 64 Mbytes. Bankx(x=1,2) is divided into two spaces, the attribute memory space and the common memory space. Bank3 is divided into three spaces, which are the attribute memory space, the common memory space and the I/O 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.

Bank1 and bank2 are used to access NAND Flash exclusively.

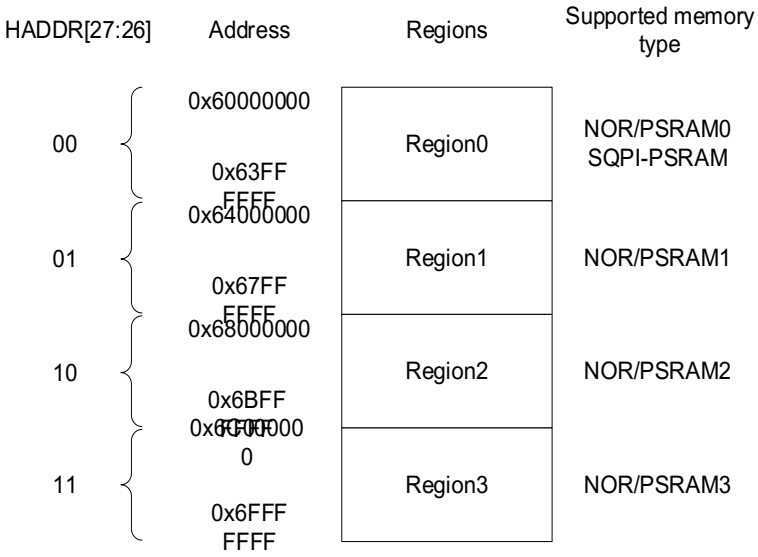
Bank3 is used for PCCard access.

SDRAM Device0 and Device1 are used for Synchronous DRAM (SDRAM) access.

NOR/PSRAM address mapping

Figure 30-3. Four regions of bank0 address mapping reflects the address mapping of the four regions of bank0. Internal AHB address lines HADDR [27:26] bits are used to select the four regions.

Figure 30-3. Four regions of bank0 address mapping



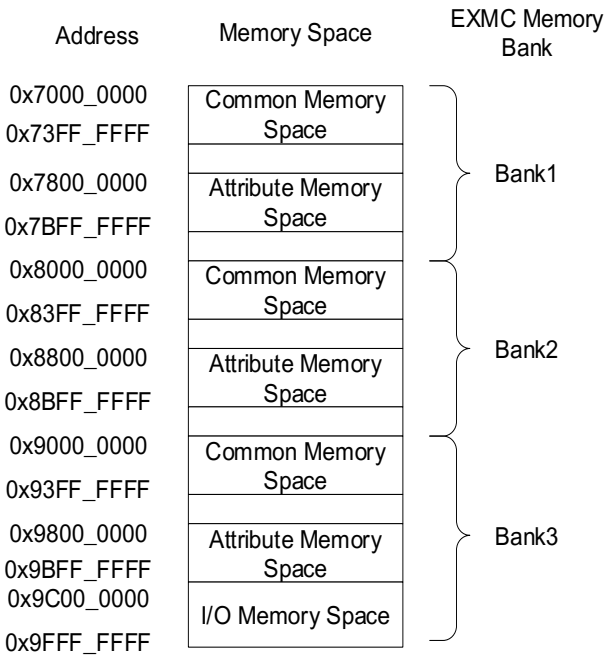
HADDR[25:0] is the byte address whereas the external memory may not be byte accessed, this will lead to address inconsistency. EXMC can adjust HADDR to accommodate the data width of the external memory according to the following rules.

- 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/PC card address mapping

Bank1 and bank2 are designed to access NAND Flash, and bank3 is designed to access PC Card. Each bank is further divided into several memory spaces as shown in ***Figure 30-4. NAND/PC card address mapping.***

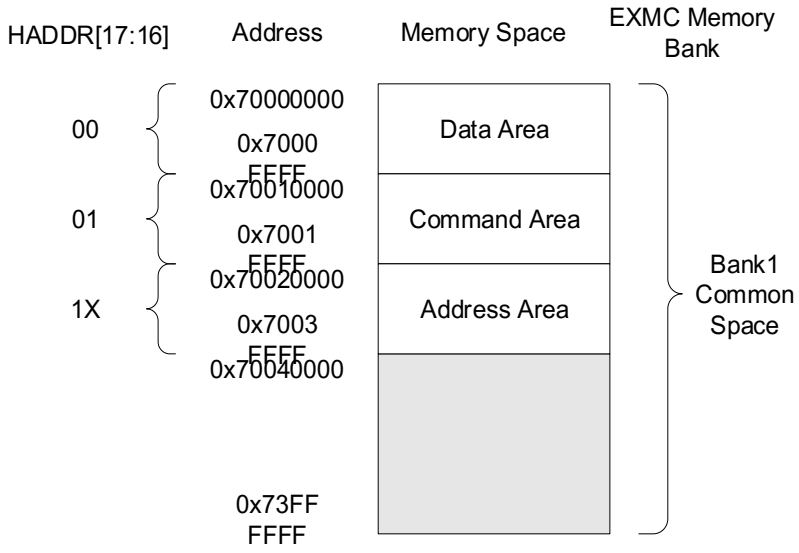
Figure 30-4. NAND/PC card address mapping



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 30-5. Diagram of bank1 common space.***

Figure 30-5. Diagram of bank1 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.

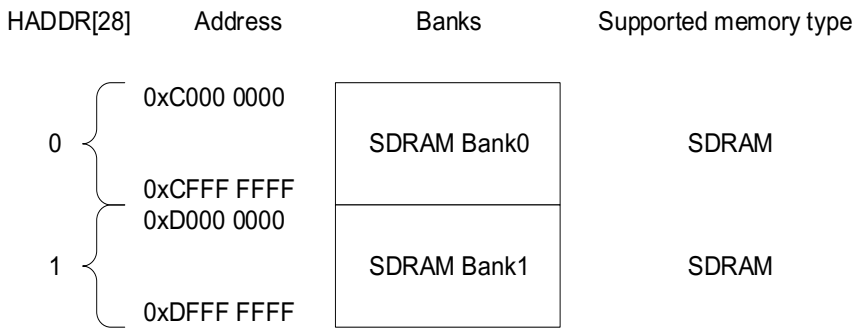
Application software uses these three areas to access NAND Flash, their definitions are as follows.

- 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 need not to be concerned with access address.

SDRAM address mapping

The HADDR [28] bit (internal AHB address line 28) is used to choose one of the two memory banks as shown in *Figure 30-6. SDRAM address mapping.*

Figure 30-6. SDRAM address mapping



The following table shows SDRAM address mapping of a 13-bit row and an 11-bit column configuration.

Table 30-1. 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

30.3.4. 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, but only sub-bank 0 support SQPI-PSRAM access, and owns its corresponding unique register.

Note:

In asynchronous mode, all output signals of controller will change on the rise edge of internal AHB bus clock (HCLK).

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 30-2. NOR flash interface signals description

EXMC Pin	Direction	Mode	Functional description
EXMC_CLK	Output	Sync	Clock signal for sync
Non-muxed EXMC_A[25:0]	Output	Async/Sync	Address bus signal
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 30-3. 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

Table 30-4. SQPI-PSRAM signal description

EXMC Pin	Direction	Mode	Function
EXMC_CLK	Output	Sync	Clock
EXMC_NE[0]	Output	Sync	Chip selection, low active
EXMC_D[0]	Input/Output	Sync	Data signal and Command signal
EXMC_D[1]	Input/Output	Sync	Data signal in SPI/SQPI/QPI mode
EXMC_D[3:2]	Input/Output	Sync	Data signal in SQPI/QPI mode

Supported memory access mode

Table below 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 30-5. EXMC bank 0 supports all transactions

Memory	Access Mode	R/W	AHB Transaction Size	Memory Transaction Size	Comments
NOR Flash	Async	R	8	16	
	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
	Sync	R	16	16	
	Sync	R	32	16	
PSRAM	Async	R	8	16	

Memory	Access Mode	R/W	AHB Transaction Size	Memory Transaction Size	Comments
	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
	Sync	R	16	16	
	Sync	R	32	16	
	Sync	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Sync	W	16	16	
	Sync	W	32	16	
SRAM and ROM	Async	R	8	8	
	Async	R	8	16	
	Async	R	16	8	Split into 2 EXMC accesses
	Async	R	16	16	
	Async	R	32	8	Split into 4 EXMC accesses
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	8	8	
	Async	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Async	W	16	8	
	Async	W	16	16	
	Async	W	32	8	
	Async	W	32	16	

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 30-6. NOR / PSRAM controller timing parameters

Parameter	Function	Access mode	Unit	Min	Max
CKDIV	Sync Clock divide ratio	Sync	HCLK	2	16
DLAT	Data latency	Sync	EXMC_CLK	2	15
BUSLAT	Bus latency	Async/Sync read	HCLK	0	15

Parameter	Function	Access mode	Unit	Min	Max
DSET	Data setup time	Async	HCLK	1	255
AHLD	Address hold time	Async(muxed)	HCLK	1	15
ASET	Address setup time	Async	HCLK	0	15

Table 30-7. EXMC_timing models

Timing model		Extend mode	Mode description	Write timing parameter	Read timing parameter
Async	Mode 1	0	SRAM/PSRAM/CRAM	DSET ASET	DSET ASET
	Mode 2	0	NOR Flash	DSET ASET	DSET ASET
	Mode A	1	SRAM/PSRAM/CRAM with EXMC_OE toggling on data phase	WDSET WASET	DSET ASET
	Mode B	1	NOR Flash	WDSET WASET	DSET ASET
	Mode C	1	NOR Flash with EXMC_OE toggling on data phase	WDSET WASET	DSET ASET
	Mode D	1	With address hold capability	WDSET WAHLD WASET	DSET AHLD ASET
	Mode AM	0	NOR Flash address/data mux	DSET AHLD ASET BUSLAT	DSET AHLD ASET BUSLAT
Sync	Mode E	0	NOR/PSRAM/CRAM synchronous read PSRAM/CRAM synchronous write	DLAT CKDIV	DLAT CKDIV
	Mode SM	0	NOR Flash address/data mux	DLAT CKDIV	DLAT CKDIV

As shown in ***Table 30-7. EXMC_timing models***, EXMC NOR Flash / PSRAM controller provides a variety of timing model, users can modify those parameters listed in ***Table 30-6. 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 30-7. Mode 1 read access

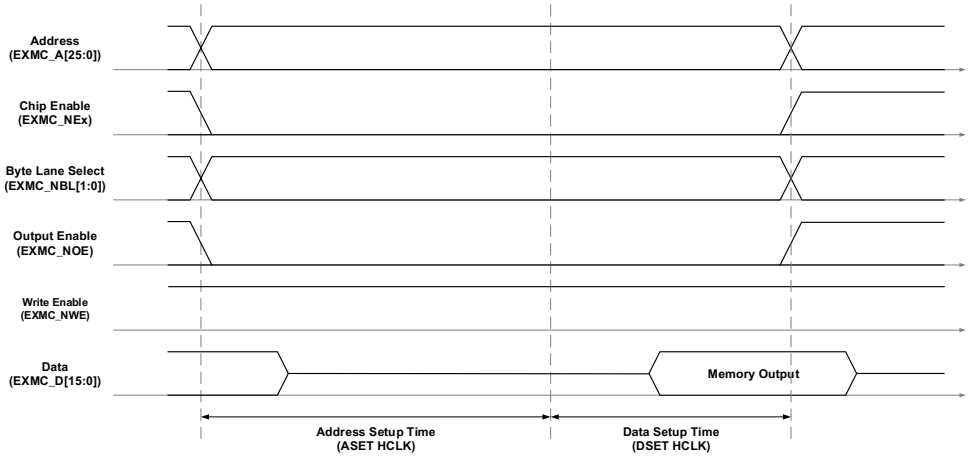


Figure 30-8. Mode 1 write access

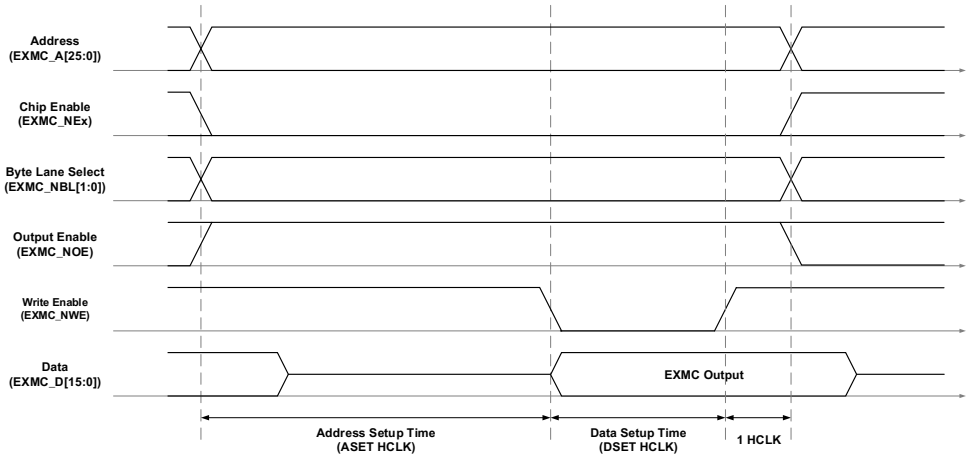


Table 30-8. Mode 1 related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory and user
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x0
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1

Bit Position	Bit Name	Reference Setting Value
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	0x0
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 HCLK for write, DSET HCLK for read)
7-4	AHLD	No effect
3-0	ASET	Depends on memory and user
EXMC_SNLATDECx		
31-3	Reserved	0x0
2-0	LATDEC	No effect

Mode A - SRAM/PSRAM(CRAM) OE toggling

Figure 30-9. Mode A read access

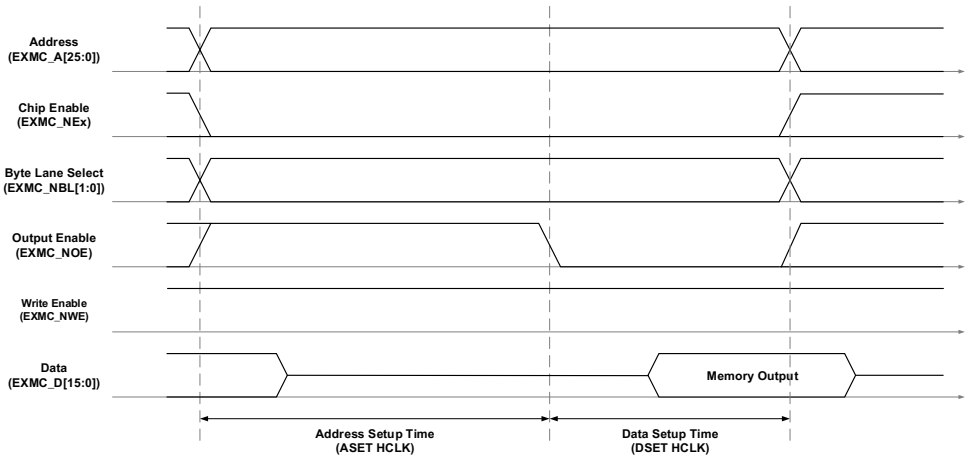
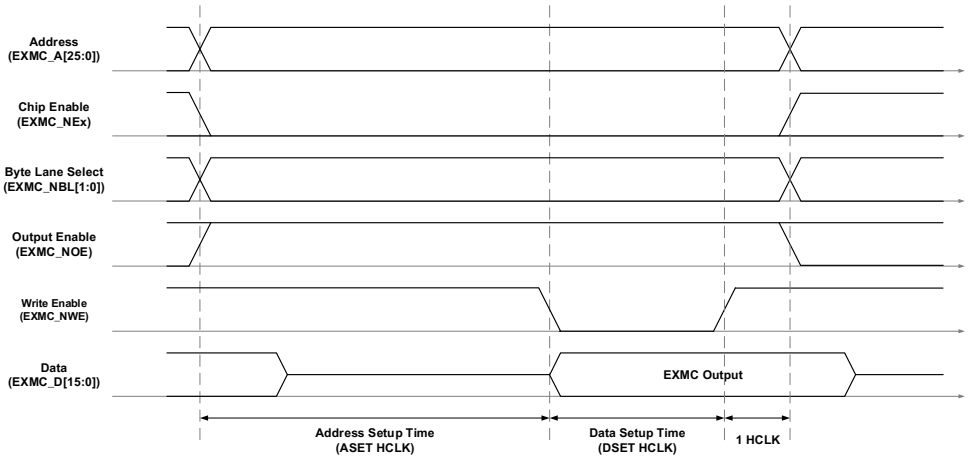


Figure 30-10. Mode A write access



The different 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 30-9. Mode A related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory and user
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	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

Bit Position	Bit Name	Reference Setting Value
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 HCLK 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 (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user
EXMC_SNLATDECx		
31-3	Reserved	0x00
2-0	LATDEC	No effect

Mode 2/B - NOR Flash

Figure 30-11. Mode 2/B read access

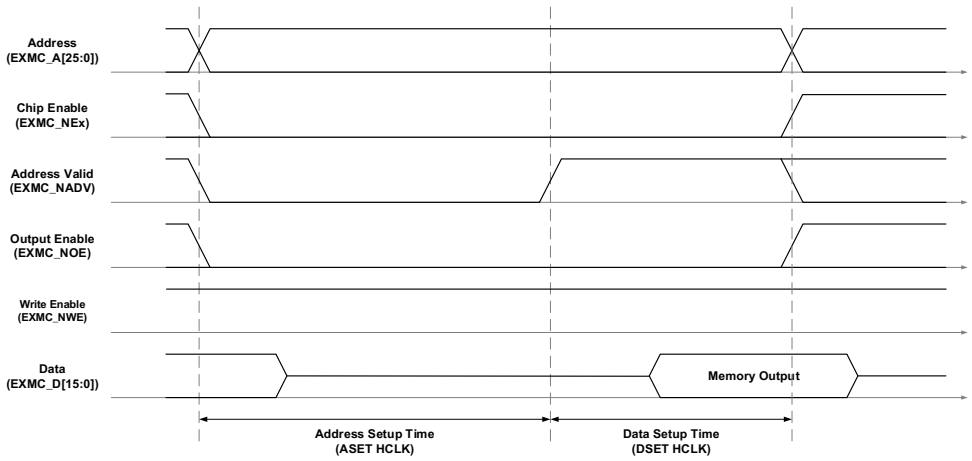


Figure 30-12. Mode 2 write access

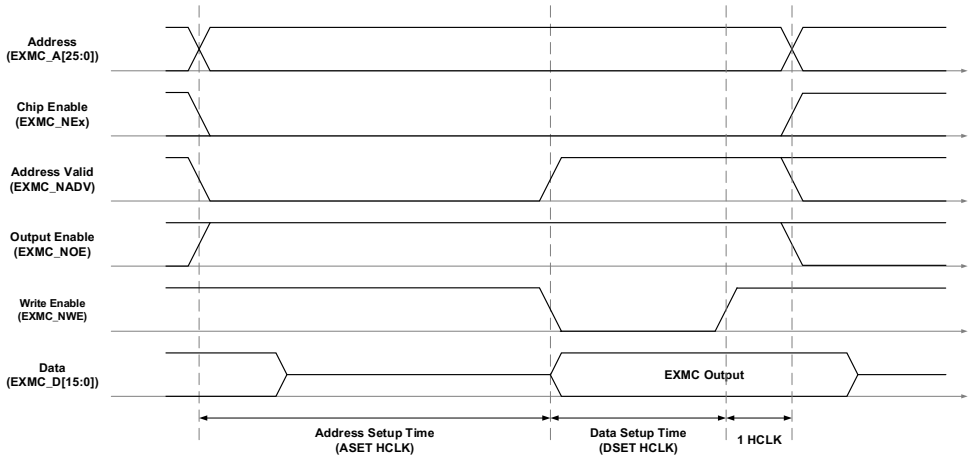


Figure 30-13. Mode B write access

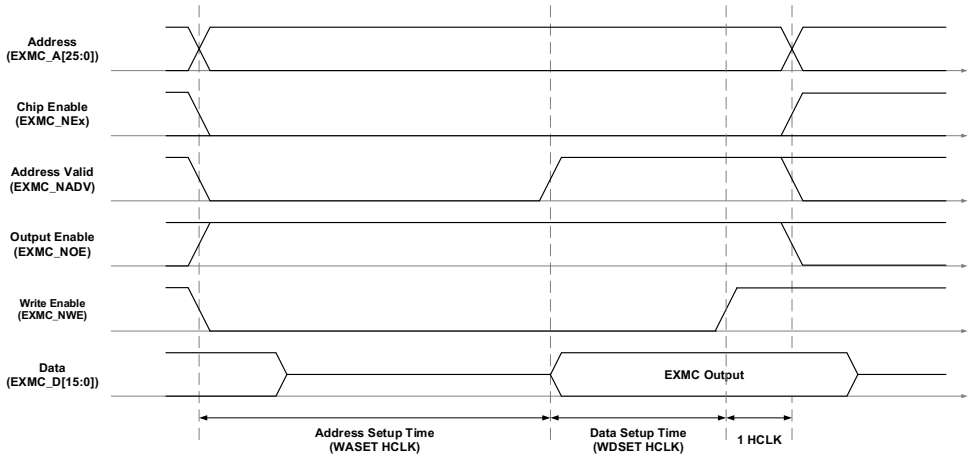


Table 30-10. Mode 2/B related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx(Mode 2, Mode B)		
31-21	Reserved	0x0
20	CCK	Depends on memory and user
19	SYNCWR	0x0
18-16	Reserved	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	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	0x1

Bit Position	Bit Name	Reference Setting Value
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	0x0
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(DSET HCLK for read, WDSET+1 HCLK for write)
7-4	AHLD	0x0
3-0	ASET	Depends on memory and user
EXMC_SNWTCFGx(Write in mode B)		
31-30	Reserved	0x0
29-28	WASYNCMOD	Mode B:0x1
27-20	Reserved	0x0
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user(WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user
EXMC_SNLATDECx		
31-3	Reserved	0x00
2-0	LATDEC	No effect

Mode C - NOR Flash OE toggling

Figure 30-14. Mode C read access

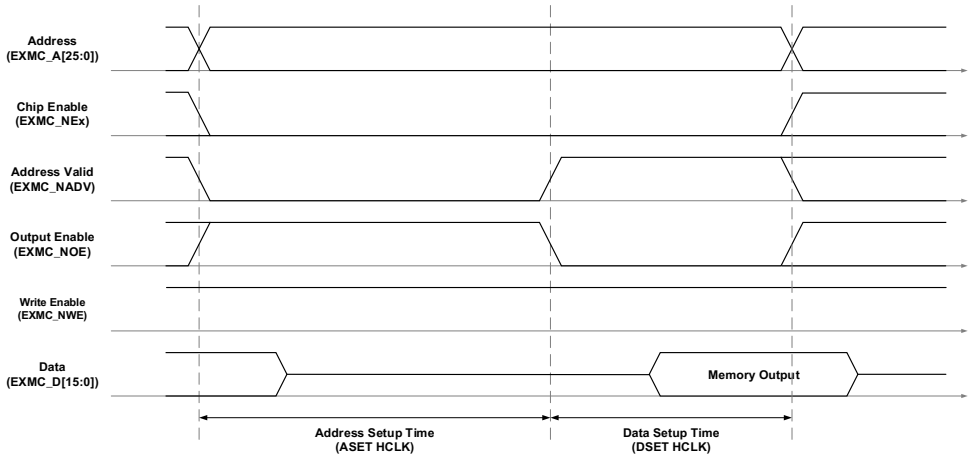
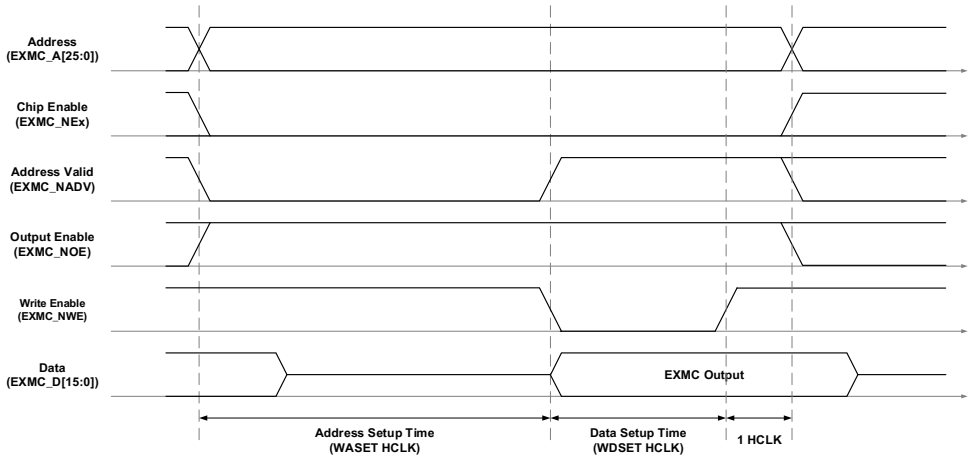


Figure 30-15. Mode C write access



The different between mode C and mode 1 write timing is that when read/write timing is specified by the same set of timing configuration, mode C write timing configuration is independent of its read configuration.

Table 30-11. Mode C related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory and user
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	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		
31-30	Reserved	0x0
29-28	ASYNCMOD	Mode C:0x2
27-24	DLAT	0x0
23-20	CKDIV	0x0

Bit Position	Bit Name	Reference Setting Value
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 HCLK for read)
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	0x0
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user(WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user
EXMC_SNLATDECx		
31-3	Reserved	0x00
2-0	LATDEC	No effect

Mode D - Asynchronous access with extended address

Figure 30-16. Mode D read access

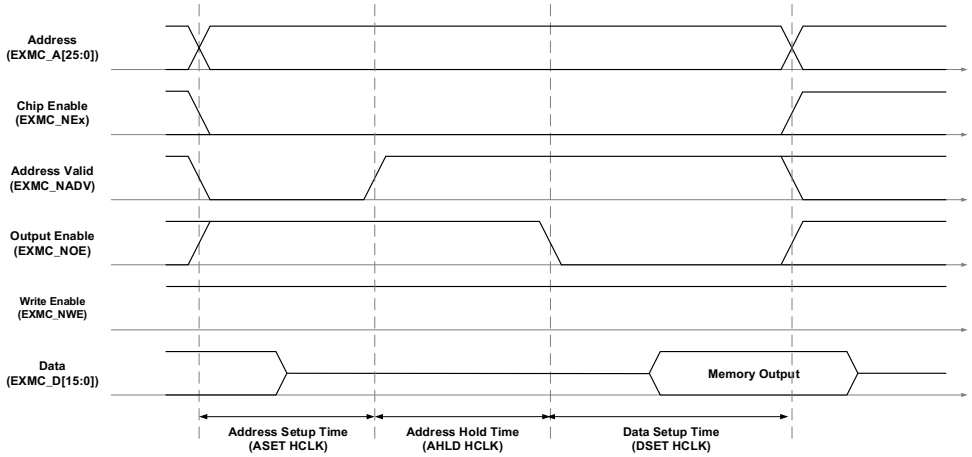


Figure 30-17. Mode D write access

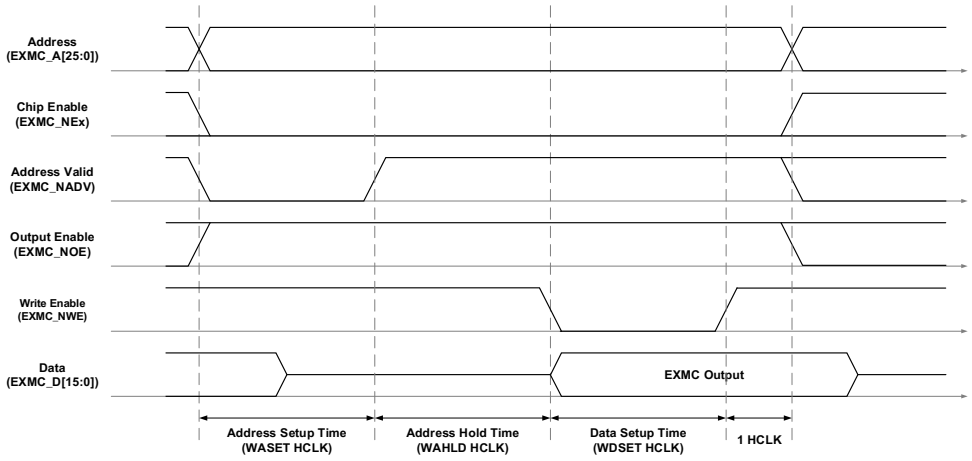


Table 30-12. Mode D related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory and user
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	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(DSET HCLK for read)

Bit Position	Bit Name	Reference Setting Value
7-4	AHLD	Depends on memory and user
3-0	ASET	Depends on memory and user
EXMC_SNWTCFGx		
31-30	Reserved	0x0
29-28	WASYNCMOD	Mode D:0x3
27-20	Reserved	0x0
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user(WDSET+1 HCLK for write)
7-4	WAHLD	Depends on memory and user
3-0	WASET	Depends on memory and user
EXMC_SNLATDECx		
31-3	Reserved	0x00
2-0	LATDEC	No effect

Mode M - NOR Flash address / data bus multiplexing

Figure 30-18. Multiplex mode read access

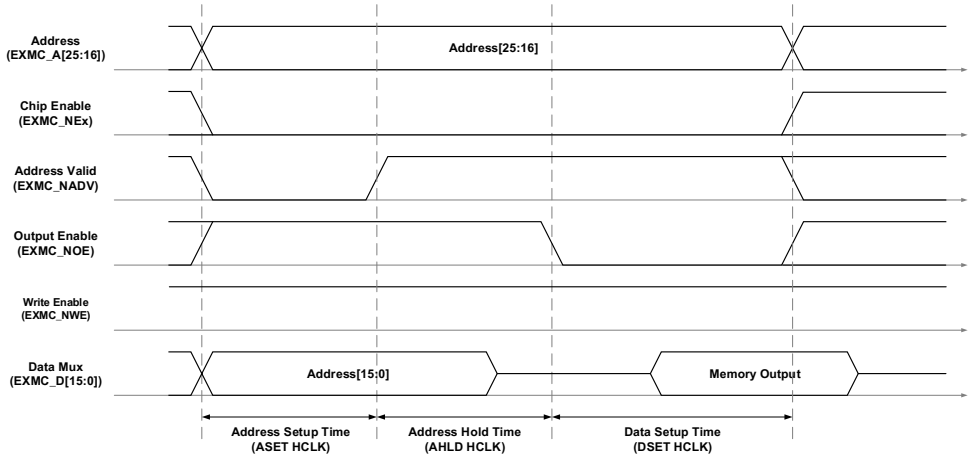


Figure 30-19. Multiplex mode write access

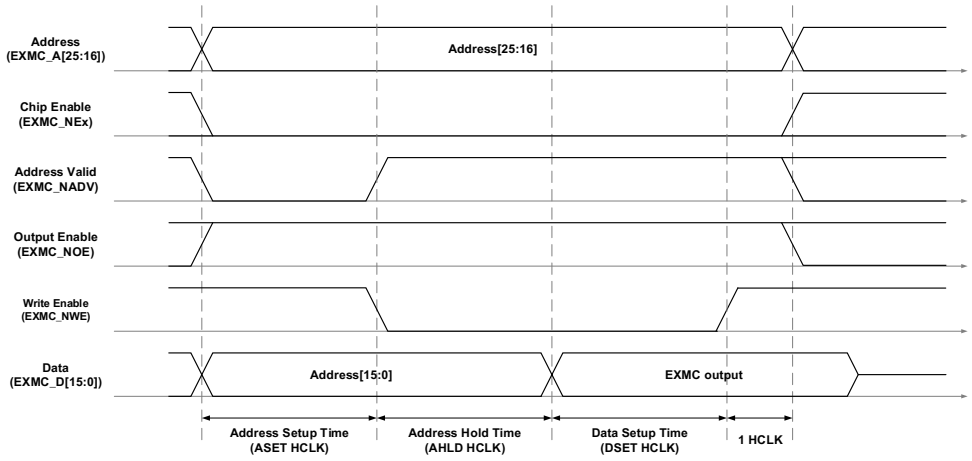


Table 30-13. Multiplex mode related registers configuration

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory
19	SYNCWR	0x0
18-16	Reserved	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x0
13	NRWTEN	0x0
12	WEN	Depends on memory
11	NRWTCFG	No effect
10	WRAPEN	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(DSET HCLK for read, WDSET+1 HCLK for write)

Bit Position	Bit Name	Reference Setting Value
7-4	AHLD	Depends on memory and user
3-0	ASET	Depends on memory and user
EXMC_SNLATDECx		
31-3	Reserved	0x00
2-0	LATDEC	No effect

Wait timing of asynchronous communication

Wait function is controlled by the bit ASYNCWTEN in register EXMC_SNCTLx. During external 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:

$$T_{DATA_SETUP} >= maxT_{WAIT_ASSERTION} + 4HCLK \quad (30\text{-}1)$$

If memory wait signal is aligned to EXMC_NE:

If

$$maxT_{WAIT_ASSERTION} >= T_{ADDRES_PHASE} + T_{HOLD_PHASE} \quad (30\text{-}2)$$

be

$$T_{DATA_SETUP} >= (maxT_{WAIT_ASSERTION} - T_{ADDRES_PHASE} - T_{HOLD_PHASE}) + 4HCLK \quad (30\text{-}3)$$

Otherwise

$$T_{DATA_SETUP} >= 4HCLK \quad (30\text{-}4)$$

Figure 30-20. Read access timing diagram under async-wait signal assertion

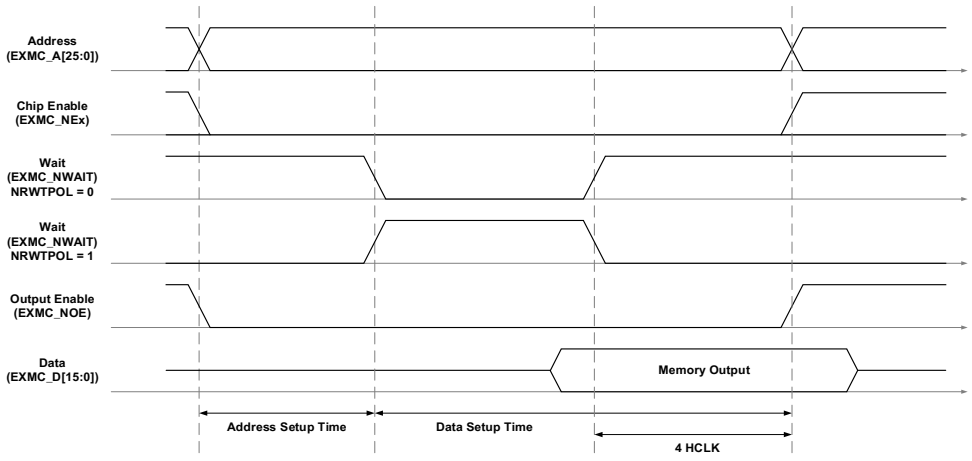
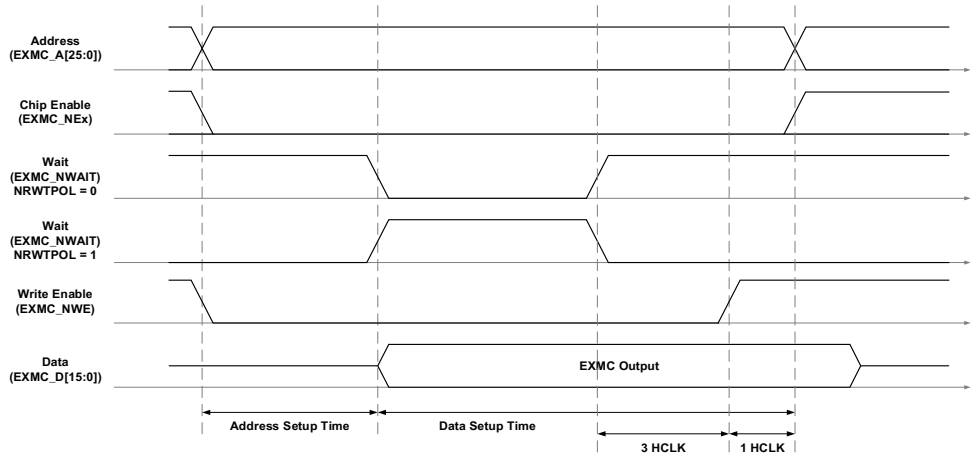


Figure 30-21. Write access timing diagram under async-wait signal assertion



Synchronous access timing diagram

The relationship between memory clock (EXMC_CLK) and system clock (HCLK) is as follows:

$$EXMC_CLK = \frac{HCLK}{CKDIV+1} \quad (30\text{-}5)$$

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 latency parameter of NOR Flash in specification is as follows.

For specification of NOR Flash excludes the EXMC_NADV cycle, their relationship should be:

$$NOR\ Flash\ latency = DLAT+2 \quad (30\text{-}6)$$

For specification of NOR Flash includes the EXMC_NADV cycle, their relationship should be:

$$NOR\ Flash\ latency = DLAT+3 \quad (30\text{-}7)$$

Note:

During read access, the data latency is determined by both DLAT in the EXMC_SNTCFGx register and LATDEC in the EXMC_SNLATDECx registers. For details, see ***SRAM/NOR flash data latency decrease registers (EXMC_SNLATDECx) (x=0, 1, 2, 3)***.

2. Data wait

Users should guarantee that EXMC_NWAIT signal matches that of the external device. This signal is configured through the EXMC_SNCTL 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 wait state by the NRWTCFG bit, and the wait signal polarity is set by the NRWTPOL bit.

In NOR Flash synchronous burst access mode, when NRWTEN bit in EXMC_SNCTL register

is set, EXMC_NWAIT signal will be detected after a period of data latency. If EXMC_NWAIT signal detected is 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 which is inserted via the EXMC_NWAIT signal, the controller continues to send clock pulses to the memory, keep the chip select signal and output signals available, 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 AHB is 16-bit, EXMC will perform a burst transmission whose length is 1. If the needed data of AHB 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 30-5. EXMC bank 0 supports all transactions***.

Read timing of synchronous multiplexed burst mode - NOR, PSRAM (CRAM)

Figure 30-22. Read timing of synchronous multiplexed burst mode

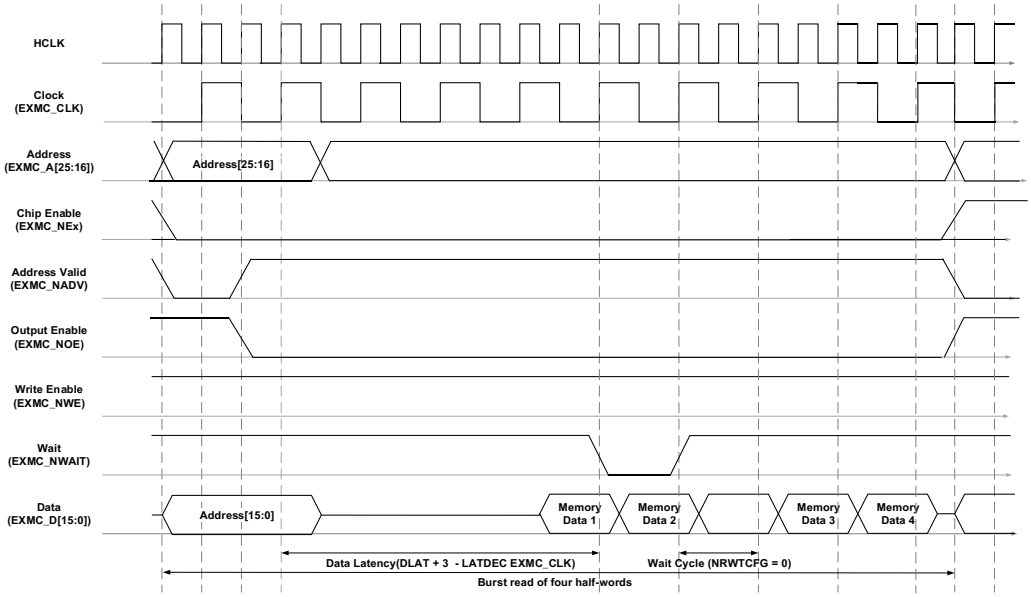


Table 30-14. Timing configurations of synchronous multiplexed read mode

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory
19	SYNCWR	No effect
18-16	Reserved	0x0
15	ASYNCWTEN	0x0
14	EXMODEN	0x0
13	NRWTEN	Depends on memory
12	WEN	0x1
11	NRWTCFG	Depends on memory
10	WRAPEN	0x0
9	NRWTPOL	Depends on memory
8	SBRSTEN	0x1, burst read enable
7	Reserved	0x1
6	NREN	0x1
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=2HCLK
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to

Bit Position	Bit Name	Reference Setting Value
		EXMC_NE[x] falling edge
15-8	DSET	No effect
7-4	AHLD	No effect
3-0	ASET	No effect
EXMC_SNLATDECx		
31-3	Reserved	0x0
2-0	LATDEC	Depends on memory and users

Mode SM - Write timing of synchronous multiplexed burst mode - PSRAM (CRAM)

Figure 30-23. Write timing of synchronous multiplexed burst mode

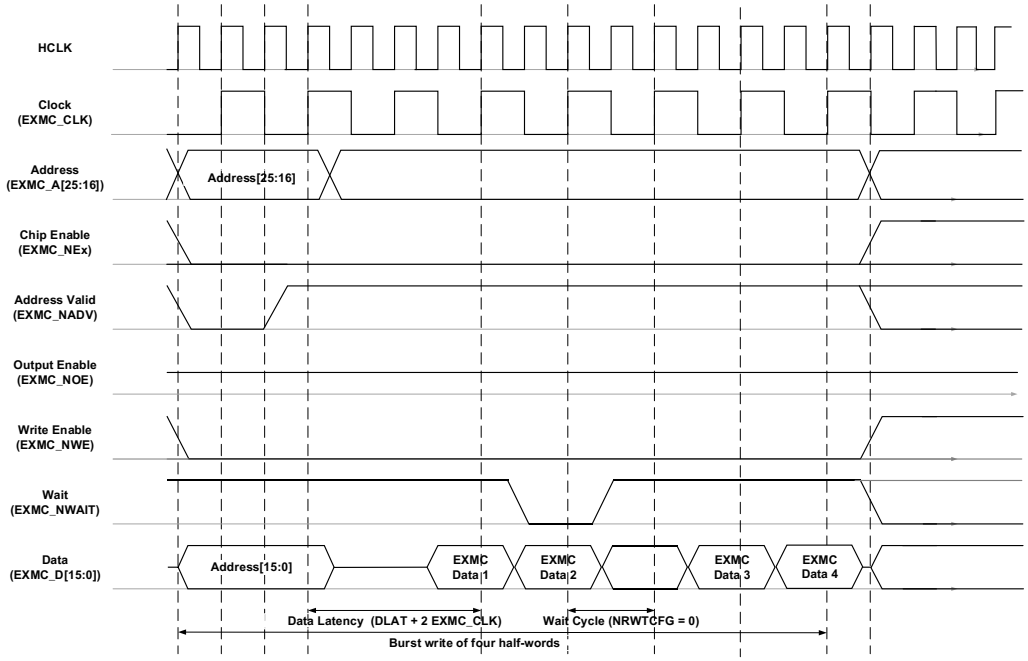


Table 30-15. Timing configurations of synchronous multiplexed write mode

Bit Position	Bit Name	Reference Setting Value
EXMC_SNCTLx		
31-21	Reserved	0x0
20	CCK	Depends on memory
19	SYNCWR	0x1, synchronous write enable
18-16	Reserved	0x0
15	AYSNCWAIT	0x0
14	EXMODEN	0x0
13	NRWTEN	Depends on memory
12	WEN	0x1
11	NRWTCFG	0x0(Here must be zero)
10	WRAPEN	0x0
9	NTWTPOL	Depends on memory
8	SBRSTEN	No effect

Bit Position	Bit Name	Reference Setting Value
7	Reserved	0x1
6	NREN	0x1
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=2HCLK
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
EXMC_SNLATDECx		
31-3	Reserved	0x0
2-0	LATDEC	No effect

SPI/QPI-PSRAM access timing diagram

SPI/QPI-PSRAM is controlled by region0 of EXMC memory bank0 only. It is a PSRAM with SPI and QPI interface, consisting of 6 IOs, the chip-enable, clock, and 4 data IOs. For more informations, please refer to the the following table.

Table 30-16. SPI/QPI interface

Signal	Direction	SPI Mode	QPI Mode
EXMC_CLK	O	Serial Clock	
EXMC_NE[0]	O	Chip-Enable (active low)	
EXMC_D[0]	IO	Serial Output	Data IO[0]
EXMC_D[1]	IO	Serial Input	Data IO[1]
EXMC_D[2]	IO	X	Data IO[2]
EXMC_D[3]	IO	X	Data IO[3]

1. Controller initialization

In the beginning, users should program the SPI initialization register EXMC_SINIT. Select the data sampling clock edge by the POL bit. Configure the read device ID length by the IDL bit. Set address bit number by the ADRBIT bit, and set command bit number by CMDBIT bit.

2. Read/Write operation

Three modes of memory access are supported, SPI, QPI, and SQPI. Access mode should be configured before read/write operation. Read/write command mode is set by the RMODE bit and WMODE bit. Wait cycle is controlled by the RWAITCYCLE bit and WWAITCYCLE bit,

and the specific memory operating command should be programmed by RCMD bit and WCMD bit. These read/write settings are located in EXMC_SRCMD and EXMC_SWCMD registers respectively.

After configuring memory access mode, read/write operation is the same as accessing ordinary NOR Flash. Data to be transferred to the external memory is written into EXMC bank0, region0, and data to be received is read from the same region.

3. Read device ID

Read device ID command is a special command. Firstly, poll the SC bit until it is 0, then set SC bit to 1. After that, the lower 32-bit ID is stored in EXMC_SIDL register, and the upper 32-bit ID is stored in EXMC_SIDH register.

4. SPI-PSRAM access timing

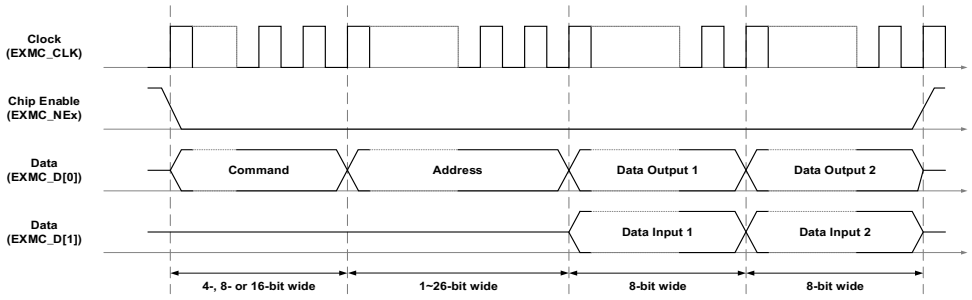
In SPI mode, the EXMC can communicates with the external memory through the SPI protocol, with 4 IOs, the clock, chip-enable, an input and an output. As shown in the diagram below, the command is first sent serially through the EXMC's data output line, which sets the external memory operating mode, and then set the address section which could be of various size depending on EXMC's configuration, and lastly, the read or write data. Data bytes are written through the data output line, while read in through the input line.

The following SPI-PSRAM waveforms are configured with:

SADRBIT[4:0] = 24,

CMDBIT[1:0] = 1.

Figure 30-24. SPI-PSRAM access



5. SQPI-PSRAM access timing

In SQPI mode, the EXMC can communicates with the external memory through the SPI protocol in command phase, and Quad SPI protocol in address and data phase with 6 IOs, the clock, chip-enable, and 4 bits data IO lines. As shown in the diagram below, the command is first sent serially through the data[0] output line, which sets the external memory operating mode, and then the 4 data IO lines output the parallel address and read/write datas.

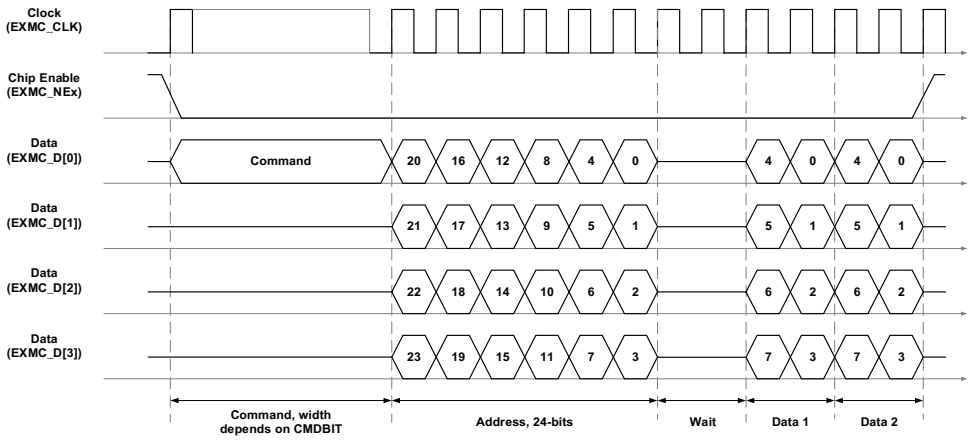
The following SQPI-PSRAM waveforms are configured with:

ADRBIT[4:0] = 24,

CMDBIT[1:0] = 1 (can be different)

RWAITCYCLE[3:0] = WWAITCYCLE[3:0] = 2 (can be different)

Figure 30-25. SQPI-PSRAM access



6. QPI-PSRAM access timing

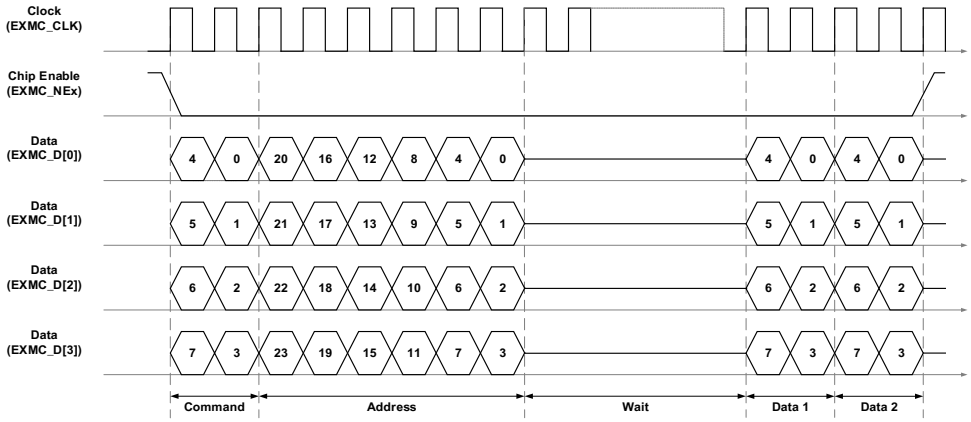
The only difference between SQPI and QPI mode is that the command is also sent parallel on the 4 data IO lines as shown in the diagram below.

The following QPI-PSRAM waveforms are configured with:

ADRBIT[4:0] = 24,

CMDBIT[1:0] = 1.

Figure 30-26. QPI-PSRAM access



30.3.5. NAND flash or PC card controller

EXMC has partitioned Bank1 and Bank2 as NAND Flash access field, bank3 as PC Card access field. Each bank has its own set of control register for access timing configuration. 8- and 16-bit NAND Flash and 16-bit PC Card are supported. An ECC hardware is provided for the NAND Flash controller to ensure the robustness of data transfer and storage.

NAND flash or PC card interface function

Table 30-17. 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[x]	Output	Chip select, x = 1, 2
EXMC_NOE(NRE)	Output	Output enable
EXMC_NWE	Output	Write enable
EXMC_NWAIT/ EXMC_INT[x]	Input	NAND Flash ready/busy input signal to the EXMC, x=1, 2

Table 30-18. 16-bit PC card interface signal

EXMC Pin	Direction	Functional description
EXMC_A[10:0]	Output	Address bus of PC Card
EXMC_NIORD	Output	I/O space output enable
EXMC_NIOWR	Output	I/O space write enable
EXMC_NREG	Output	Register signal indicating if access is in Common space or Attribute space
EXMC_D[15:0]	Input /Output	Bidirectional data bus
EXMC_NCE3_x	Output	Chip select(x=0,1)
EXMC_NOE	Output	Output enable
EXMC_NWE	Output	Write enable
EXMC_NWAIT	Input	PC Card wait input signal to the EXMC
EXMC_INTR	Input	PC Card interrupt input signal
EXMC_CD	Input	PC Card presence detection. Active high.

Supported memory access mode

Table 30-19. Bank1/2/3 of EXMC support the memory and access mode

Memory	Mode	R/W	AHB 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	
16-bit NAND/PC Card	Async	R	8	
	Async	W	8	Not support this operation
	Async	R	16	
	Async	W	16	

Memory	Mode	R/W	AHB transaction size	Comments
	Async	R	32	Automatically split into 2 EXMC accesses
	Async	W	32	

NAND flash or PC card controller timing

EXMC can generate the appropriate signal timing for NAND Flash, PC Cards and other devices. Each bank has a corresponding register to manage and control the external memory, such as EXMC_NPCTLx, EXMC_NPINTENx, EXMC_NPCTCFGx, EXMC_NPATCFGx, EXMC_PIOTCFG3 and EXMC_NECCx. Among these registers, EXMC_NPCTCFGx, EXMC_NPATCFGx, EXMC_PIOTCFG3 can configure four timing parameters individually according to user specification and features of the external memory.

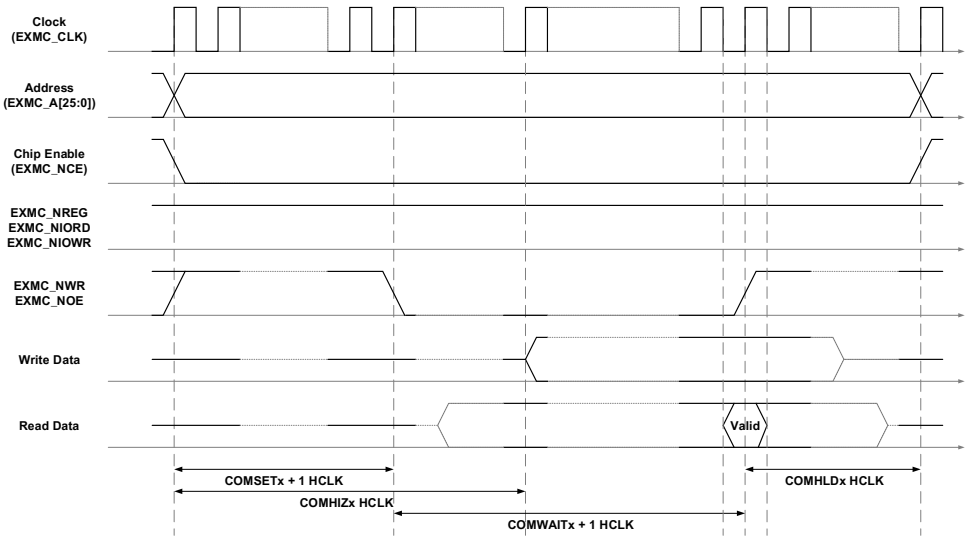
Table 30-20. NAND flash or PC card programmable parameters

Programmable parameter	W/R	Unit	Functional description	NAND Flash/ PC Card	
				Min	Max
High impedance time of the memory data bus (HIZ)	W/R	HCLK	Time to keep the data bus high impedance after starting write operation	1	255
Memory hold time(HLD)	W/R	HCLK	The number of HCLK 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	HCLK	Minimum duration of sending command	2	255
Memory setup time(SET)	W/R	HCLK	The number of HCLK clock cycles to build address before sending command	1	256

The figure below shows the programmable parameters which are defined in the common memory space operations. The programmable parameters of Attribute memory space or I/O memory space (only for PC Card) are defined as well.

Figure 30-27. Access timing of common memory space of NAND flash or PC card

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:

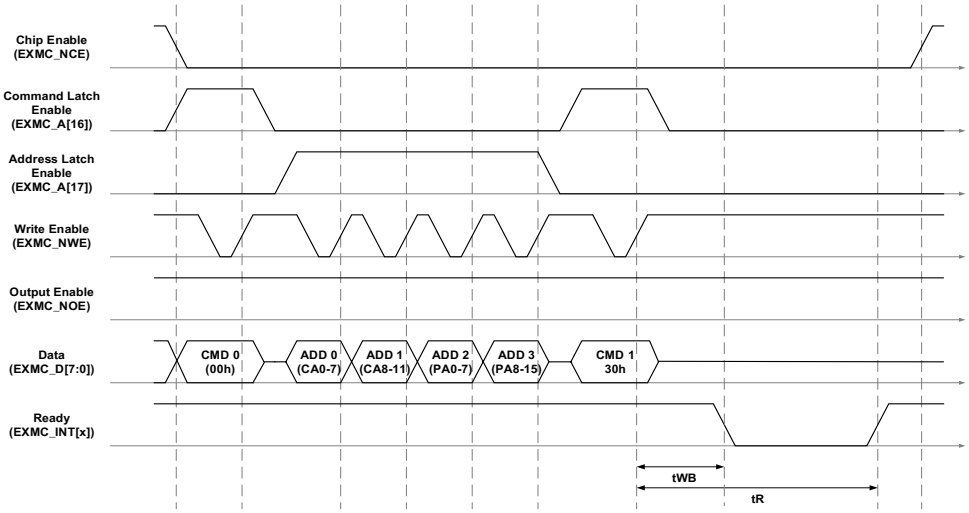
1. Configure EXMC_NPCTLx and EXMC_NPCTCFGx register. When pre-waiting is needed, EXMC_NPATCFGx has to be configured.
2. Send the command of NAND Flash read operation to the common space. Namely, during the valid period of EXMC_NCE and EXMC_NWE, when EXMC_CLE (EXMC_A [16]) becomes valid (high level), 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 command or address hasn'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 then writing a new address, or go to step2, writing a new command and address.

NAND Flash pre-wait functionality

Some NAND Flash requires that the controller should wait for NAND Flash to be busy after the first command byte following the address bytes is send. 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 30-28. 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_NPATCFGx register. After a period of ATTHLD, NAND Flash waits for EXMC_INTx 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_INTx low). For NCE-sensitive NAND Flash, after the first command byte following address bytes has been entered, EXMC_NCE must remain low until EXMC_INTx goes from low to high. The ATTHLD value of attribute space can be set in EXMC_NPATCFGx register to meet the timing requirements of tWB. MCU can use the attribute space timing when writing thefirst 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 bank1 and bank2 respectively. Users can choose page size according to the ECCSZ control field in the EXMC_NPCTLx 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 theEXMC_NECCx register. When a recalculation of ECC is needed, software must clear the EXMC_NECCx register value by resetting ECCEN bit of EXMC_NPCTLx register to 0, and then restart ECC calculation by setting the ECCEN bit of EXMC_NPCTLx to 1.

PC/CF card access

EXMC Bank3 is used exclusively for PC/CF Card, both memory and IO mode access are supported. This bank is divided further into three sub spaces, memory, attribute and IO space.

EXMC_NCE3_0 and EXMC_NCE3_1 are the byte select signals, when only EXMC_NCE3_0 is active (Low), the lower byte or upper byte is selected depending on the EXMC_A[0], while only EXMC_NCE3_1 is active (Low), the upper byte is selected which is not supported, when both of these signals are active, 16-bit operation is performed. When NDTP is reset to select PC/CF Card as external memory device, NDW must be set to 01 in EXMC_NPCTLx register to guarantee correct EXMC operation.

EXMC PC/CF card access behavior for different spaces:

1. Common space: EXMC_NCE3_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC_NWE and EXMC_NOE dictates whether the on-going operation is a write or read operation, and EXMC_NREG is high during common space access.
2. Attribute space: EXMC_NCE3_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC_NWE and EXMC_NOE dictates whether the on-going operation is a write or read operation, and EXMC_NREG is low during attribute space access.
3. IO space: EXMC_NCE3_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC_NIOWR and EXMC_NIORD dictates whether the on-going operation is a write or read operation, and EXMC_NREG is low during IO space access.

AHB access on 16-bit PC/CF card:

1. Common space: It is usually where data are stored, it could be accessible either in byte or in half-word mode, and odd address access is not supported in byte mode. When AHB word access is selected, EXMC automatically splits it into 2 consecutive half-word access. EXMC_NREG is high when common memory is targeted. EXMC_NOE and EXMC_NWE are the read and write enable signal for this type of access.
2. Attribute space: It is usually where configuration information are stored, for byte AHB access, only even address is possible. Half-word access converts into a single byte access automatically, and word access is converted into two consecutive byte access where only the even bytes are operational. In both half-word and word access, only EXMC_NCE3_0 will be active. EXMC_NREG is low when attribute memory is targeted. EXMC_NOE and EXMC_NWE are the read and write enable signal for this type of access.
3. IO space: Both byte and half-word AHB access are supported, in IO space memory access, EXMC_NIORD and EXMC_NIOWR act as the read and write enable signal respectively.

30.3.6. 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)
- AHB 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 $f_{HCLK}/2$ or f_{HCLK} /3
- 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). Its external interface is coordinated by a synchronous external clock, which is provided by the EXMC through the SDRAM clock (EXMC_SDCLK) pin and can be configured frequency to be $f_{HCLK}/2$ or f_{HCLK} /3 according to the SDRAM clock configuration bit (SDCLK) in the EXMC_SDCTLx register. Command and data are always latched by the SDRAM on the rising edge of EXMC_SDCLK and changes 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.

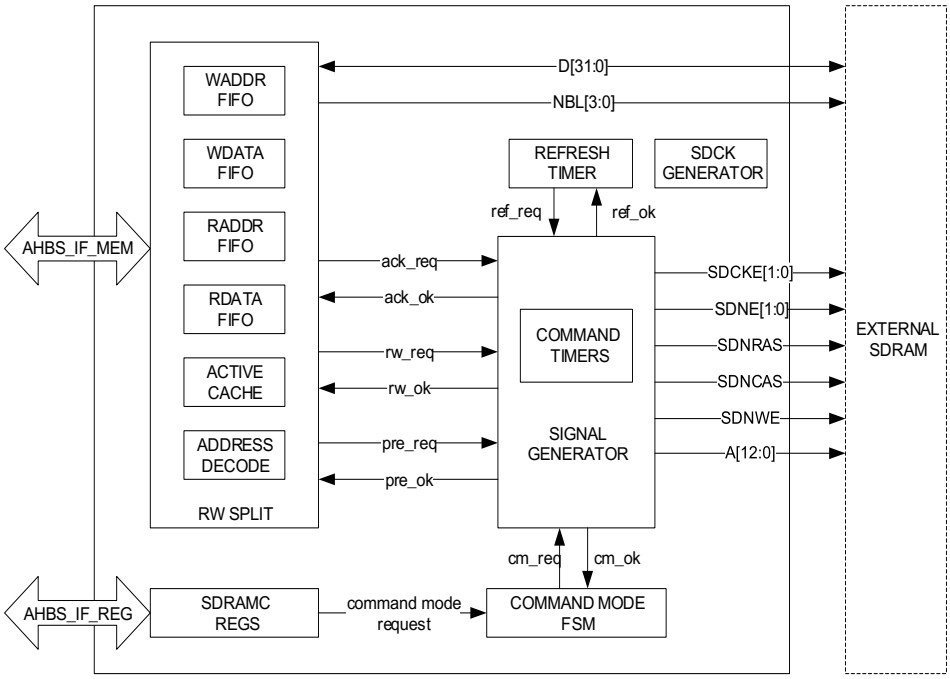
Mode Register it is used to define the specific operating mode of SDRAM, such modes include burst length, burst type, CAS latency, and write mode. Users should refer to the SDRAM's specification for correct configuration. Once the operating mode has been decided, users should write the mode register 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 AHB 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 in to 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 ***Figure 30-29. SDRAM controller block diagram*** is shown as follows.

Figure 30-29. 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 ***Table 30-21. SDRAM command truth table***.

Table 30-21. 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
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 Self-refresh when SDCKE = 0

SDNE	NRAS	NCAS	SDNWE	A[n]	A[10]	A[m]	Command
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 the following table *Table 30-22. IO definition of SDRAM controller*.

Table 30-22. 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 whichSDRAM 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.

9. This SDRAMC is ready to proceed with memory access at this stage, 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 banks are 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.

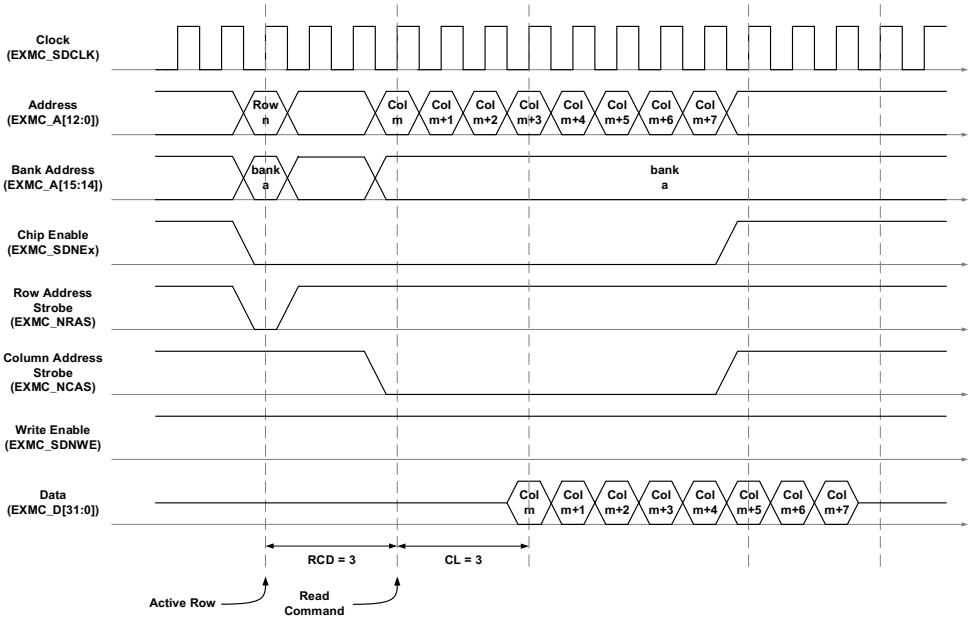
Once the row has been activated, read/write commands are possible to that row. Activation requires a minimum amount of time, called the row-to-column delay (RCD) before read/write to it may occur. This time, rounded up to the next multiple of the clock period, specifies the minimum number of wait cycles 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 AHB 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 following diagram *Figure 30-30. 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.

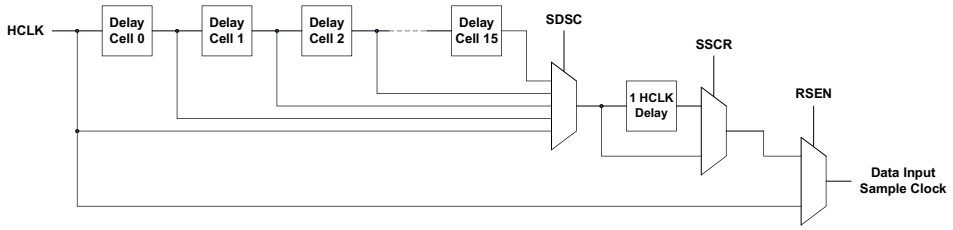
Figure 30-30. Burst read operation



An internal generated clock, which has an adjustable delay from the HCLK 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 HCLK. When this clock is enabled, the read data will be firstly stored in an asynchronous FIFO before returned to the AHB bus. Additional delays of about 2~3 HCLK may be brought into the reading command process.

A clock delay chain module is added after the HCLK 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 HCLK output is delay at all, SSCR bit select whether 1 additional HCLK 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.

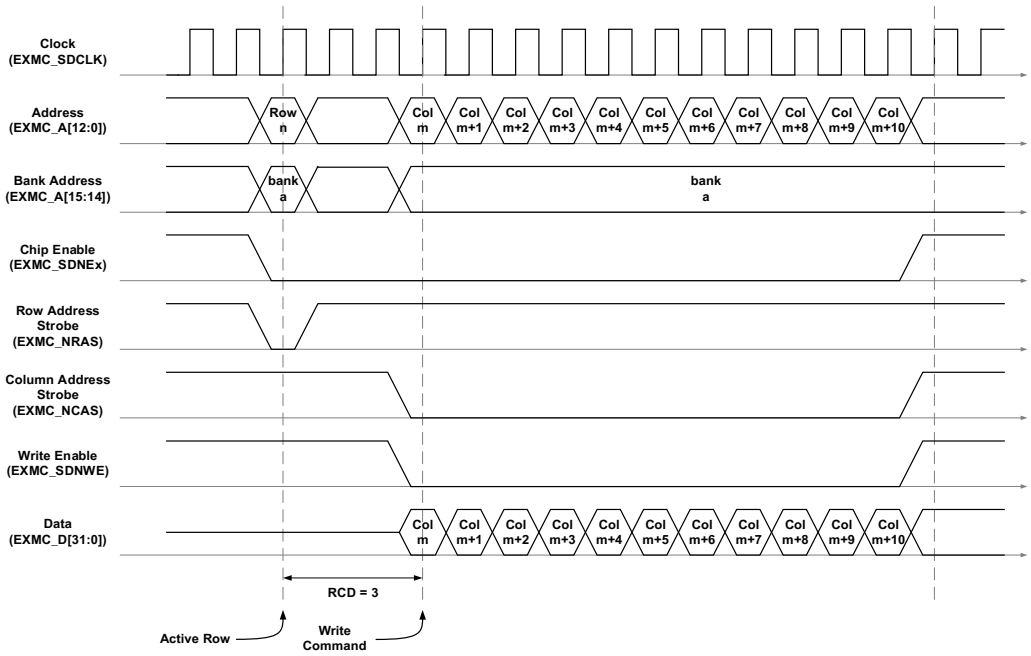
Figure 30-31. Data sampling clock delay chain



SDRAMC can translate AHB 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 following diagram ***Figure 30-32. 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 30-32. Burst write operation



The RW split module accepts AHB commands, and transfers them to single read/write accesses on the SDRAM memory according to the ratio of the data width between the AHB bus and the SDRAM memory interface.

Inside the RW split module, there are two write FIFOs, which buffers the data and address of the AHB 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 buffer 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 AHB 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 30-33. Read access when FIFO not hit (BRSTRD=1, CL=2, SDCLK=2, PIPED=2) and *Figure 30-34. Read access when FIFO hit (BRSTRD=1)* specify the Read FIFO operation.

Figure 30-33. Read access when FIFO not hit (BRSTRD=1, CL=2, SDCLK=2, PIPED=2)

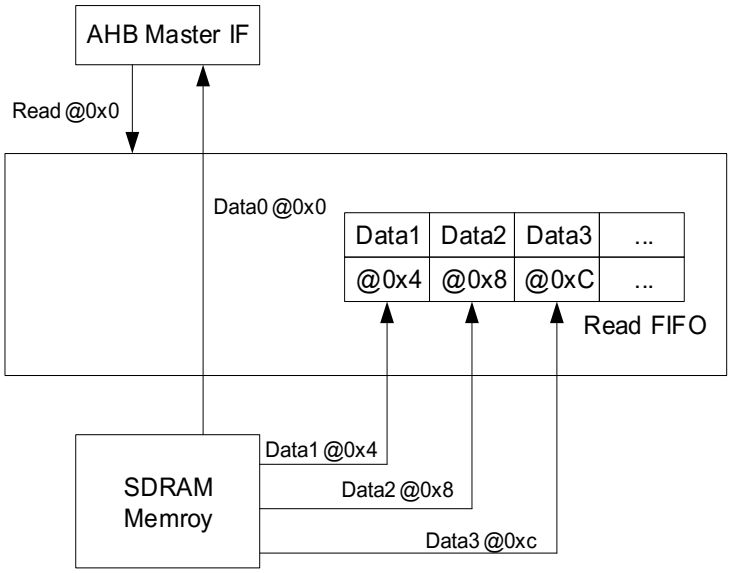
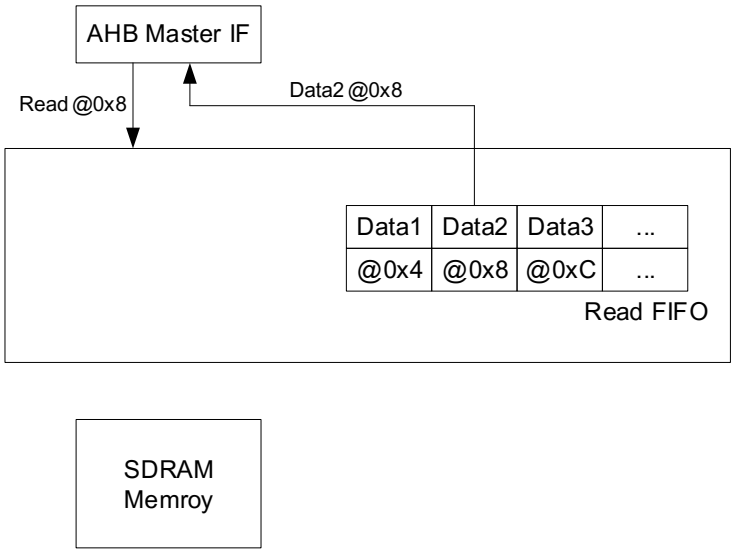


Figure 30-34. Read access when FIFO hit (BRSTRD=1)



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 AHB 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 AHB 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 30-35. Cross boundary read operation

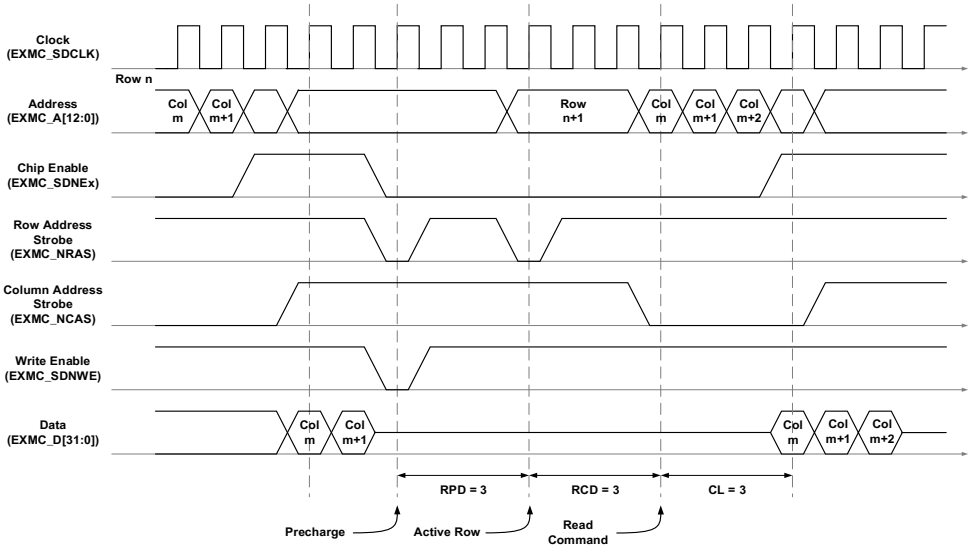
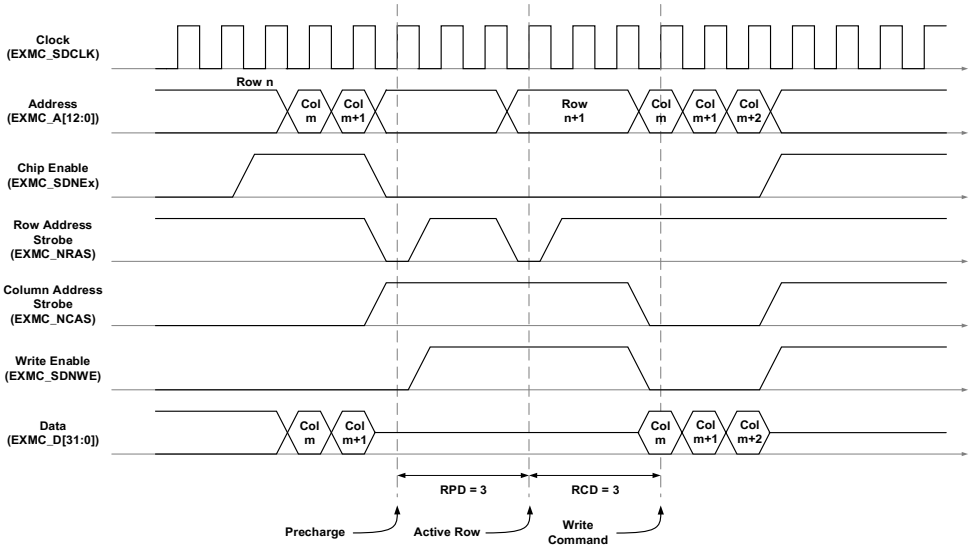


Figure 30-36. Cross boundary write operation



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

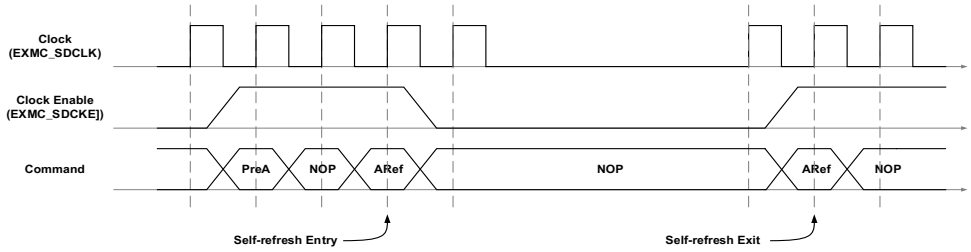
Two low power mode are supported:

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).

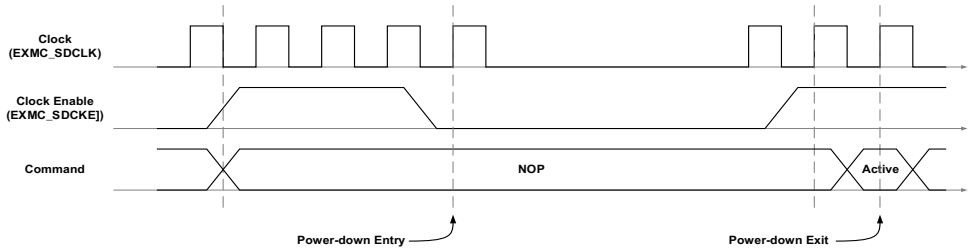
The SDRAM controller returns to normal mode from self-refresh mode when a read/write access occurs. If a read/write access occurs while the SDRAM controller is entering self-refresh mode, the self-refresh entry process will be interrupted, and the SDRAM controller remains in normal mode after the read/write access completed.

Figure 30-37. 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 30-38. 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.

30.4. Register definition

EXMC base address: 0xA000 0000

30.4.1. NOR/PSRAM controller registers

SRAM/NOR flash control registers (EXMC_SNCTLx) (x=0, 1, 2, 3)

Address offset: 0x00 + 8 * x, (x = 0, 1, 2, 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											CCK	SYNCWR	CPS[2:0]		
											rw	rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNC WTEN	EXMO DEN	NRWT EN	WEN	NRWT CFG	WRAPEN	NRWT POL	SBR STEN	Reserved	NREN	NRW[1:0]		NRTP[1:0]		NR MUX	NRBK EN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw		rw	rw

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20	CCK	Consecutive clock 0: EXMC_CLK is generated only during synchronous access. 1: EXMC_CLK is generated unconditionally. **Note**: Consecutive clock (CCK) bit is only present in EXMC_SNCTL0 register, and this bit position inside EXMC_SNCTLx (x = 1, 2, 3) registers is mearningless.
19	SYNCWR	Synchronous write 0: Asynchronous write 1: Synchronous write
18:16	CPS[2:0]	CRAM page size 000: Automatic burst split on page boundary crossing 001: 128 bytes 010: 256 bytes 011: 512 bytes 100: 1024 bytes Others: Reserved
15	ASYNCWTEN	Asynchronous wait enable 0: Disable the asynchronous wait function 1: Enable the asynchronous wait function
14	EXMODEN	Extended mode enable

		0: Disable extended mode 1: Enable extended mode
13	NRWTEN	NWAIT signal enable For flash memory access in burst mode, this bit enables/disables wait-state insertion to the NWAIT signal. 0: Disable NWAIT signal 1: Enable NWAIT signal
12	WEN	Write enable 0: Disable write in the bank by the EXMC, otherwise an AHB error is reported 1: Enable write in the bank by the EXMC (default after reset)
11	NRWTCFG	NWAIT signal configuration, only work in synchronous mode 0: NWAIT signal is active one data cycle before wait state 1: NWAIT signal is active during wait state
10	WRAPEN	Wrapped burst mode enable 0: Disable wrap burst mode support 1: Enable wrap burst mode support
9	NRWTPOL	NWAIT signal polarity 0: Low level of NWAIT is active 1: High level of NWAIT is active
8	SBRSTEN	Synchronous burst enable 0: Disable burst access mode 1: Enable burst access mode
7	Reserved	Must be kept at reset value.
6	NREN	NOR Flash access enable 0: Disable NOR Flash access 1: Enable NOR Flash access
5:4	NRW[1:0]	NOR region memory data bus width 00: 8 bits 01: 16 bits(default after reset) 10/11: Reserved
3:2	NRTP[1:0]	NOR region memory type 00: SRAM, ROM 01: PSRAM(CRAM) 10: NOR Flash 11: Reserved
1	NRMUX	NOR region memory address/data multiplexing 0: Disable address/data multiplexing function 1: Enable address/data multiplexing function

0	NRBKEN	NOR region enable 0: Disable the corresponding memory bank 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, 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 The bits are valid only when the EXMEN bit in the EXMC_SNCTLx register is 1. 00: Mode A access 01: Mode B access 10: Mode C access 11: Mode D access
27:24	DLAT[3:0]	Data latency for NOR Flash. Only valid in synchronous access 0x0: Data latency of first burst access is 2 EXMC_CLK 0x1: Data latency of first burst access is 3 EXMC_CLK …… 0xD: Data latency of first burst access is 15 EXMC_CLK 0xE ~ 0xF: Reserved
23:20	CKDIV[3:0]	Synchronous clock divide ratio. This filed is only effect in synchronous mode. 0x0: Reserved 0x1: EXMC_CLK period = 2 * HCLK period …… 0xF: EXMC_CLK period = 16 * HCLK period
19:16	BUSLAT[3:0]	Bus latency 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. 0x0: Bus latency = 0 * HCLK period 0x1: Bus latency = 1 * HCLK period ……

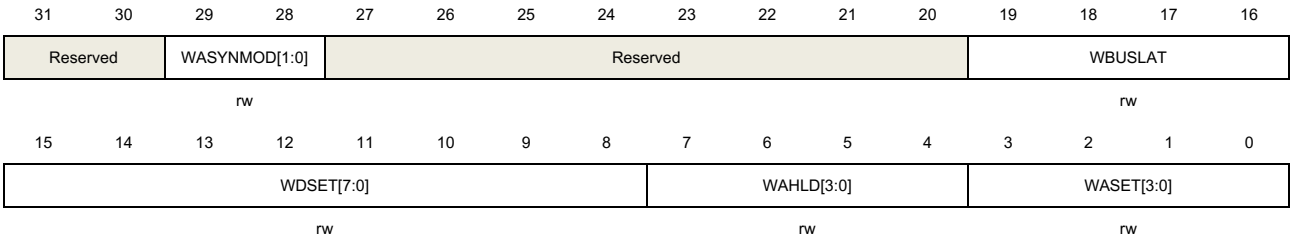
		0xF: Bus latency = 15 * HCLK period
15:8	DSET[7:0]	Data setup time This field is meaningful only in asynchronous access. 0x00: Reserved 0x01: Data setup time = 1 * HCLK period …… 0xFF: Data setup time = 255 * HCLK period
7:4	AHLD[3:0]	Address hold time This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode. 0x0: Reserved 0x1: Address hold time = 1 * HCLK …… 0xF: Address hold time = 15 * HCLK
3:0	ASET[3:0]	Address setup time This field is used to set the time of address setup phase. **Note:** meaningful only in asynchronous access of SRAM, ROM, NOR Flash 0x0: Address setup time = 0 * HCLK …… 0xF: Address setup time = 15 * HCLK

SRAM/NOR flash write timing configuration registers (EXMC_SNWTCFGx) (x=0, 1, 2, 3)

Address offset: 0x104 + 8 * x, (x = 0, 1, 2, 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).



Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:28	WASYNCMOD[1:0]	Asynchronous access mode The bits are valid only when the EXMEN bit in the EXMC_SNCTLx register is 1. 00: Mode A access 01: Mode B access

		10: Mode C access 11: Mode D access
27:20	Reserved	Must be kept at reset value.
19:16	WBUSLAT[3:0]	Bus latency Bus latency added at the end of each write transaction to match with the minimum time between consecutive transactions. 0x0: Bus latency = 0 * HCLK period 0x1: Bus latency = 1 * HCLK period …… 0xF: Bus latency = 15 * HCLK period
15:8	WDSET[7:0]	Data setup time This field is meaningful only in asynchronous access. 0x00: Reserved 0x01: Data setup time = 1 * HCLK period …… 0xFF: Data setup time = 255 * HCLK period
7:4	WAHLD[3:0]	Address hold time This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode. 0x0: Reserved 0x1: Address hold time = 1 * HCLK …… 0xF: Address hold time = 15 * HCLK
3:0	WASET[3:0]	Address setup time This field is used to set the time of address setup phase. **Note:** Meaningful only in asynchronous access of SRAM,ROM,NOR Flash 0x0: Address setup time = 0 * HCLK 0x1: Address setup time = 1 * HCLK …… 0xF: Address setup time = 15 * HCLK

SRAM/NOR flash data latency decrease registers (EXMC_SNLATDECx) (x=0, 1, 2, 3)

Address offset: 0x300+ 4 * x, (x= 0, 1, 2, and 3)
Reset value: 0x0000 0000

This register is meaningful only in synchronous access.

This register 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													LATDEC[2:0]		
													rw		

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	LATDEC[2:0]	Data latency decrease for NOR Flash. Only valid in synchronous read access. This field is used to adjust read access time along with DLAT. Burst read: 000: Data latency of first burst access is (DLAT + 3) EXMC_CLK 001: Data latency of first burst access is (DLAT + 2) EXMC_CLK 010: Data latency of first burst access is (DLAT + 1) EXMC_CLK 011: Data latency of first burst access is (DLAT + 0) EXMC_CLK 100: Data latency of first burst access is (DLAT - 1) EXMC_CLK 101: Data latency of first burst access is (DLAT - 2) EXMC_CLK 110: Data latency of first burst access is (DLAT - 3) EXMC_CLK 111: Data latency of first burst access is (DLAT - 4) EXMC_CLK **Note:** For example, if the data latency in read mode needs to be configured with 3 CLK, the DLAT[3:0] should be 0b'0000 and LATDEC[2:0] should be 0b'010.

30.4.2. NAND flash/PC card controller registers

NAND flash/PC card control registers (EXMC_NPCTLx) (x=1, 2, 3)

Address offset: 0x40 + 0x20 * x, (x = 1, 2, and 3)

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												ECCSZ[2:0]			ATR[3]
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATR[2:0]			CTR[3:0]				Reserved		ECCEN	NDW[1:0]		NDTP	NDBKEN	NDWTEN	Reserved
rw			rw						rw	rw		rw	rw	rw	

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:17	ECCSZ[2:0]	ECC size 000: 256 bytes 001: 512 bytes 010: 1024 bytes 011: 2048 bytes

		100: 4096 bytes 101: 8192 bytes
16:13	ATR[3:0]	ALE to RE delay 0x0: ALE to RE delay = 1 * HCLK …… 0xF: ALE to RE delay = 16 * HCLK
12:9	CTR[3:0]	CLE to RE delay 0x0: CLE to RE delay = 1 * HCLK 0x1: CLE to RE delay = 2 * HCLK …… 0xF: CLE to RE delay = 16 * HCLK
8:7	Reserved	Must be kept at reset value.
6	ECCEN	ECC enable 0: Disable ECC, and reset EXMC_NECCx 1: Enable ECC
5:4	NDW[1:0]	NAND bank memory data bus width 00: 8 bits 01: 16 bits Others: Reserved **Note**: for PC/CF card, 16-bit bus width must be selected.
3	NDTP	NAND bank memory type 0: PC Card, CF card, PCMCIA 1: NAND Flash
2	NDBKEN	NAND bank enable 0: Disable corresponding memory bank 1: Enable corresponding memory bank
1	NDWTEN	Wait function enable 0: Disable wait function 1: Enable wait function
0	Reserved	Must be kept at reset value.

NAND flash/PC card interrupt enable registers (EXMC_NPINTENx) (x=1, 2, 3)

Address offset: 0x44 + 0x20 * x, (x = 1,2, and 3)

Reset value: 0x0000 0042 (for bank1 and bank2), 0x0000 0040 (for bank3)

This register 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 0: FIFO is not empty. 1: FIFO is empty.
5	INTFEN	Falling edge detection interrupt enable 0: Disable falling edge detection interrupt 1: Enable falling edge detection interrupt
4	INTHEN	High-level detection interrupt enable 0: Disable high-level detection interrupt 1: Enable high-level detection interrupt
3	INTREN	Rising edge detection interrupt enable 0: Disable rising edge detection interrupt 1: Enable rising edge detection interrupt
2	INTFS	Falling edge flag 0: Not detect falling edge 1: Detect falling edge
1	INTHS	High-level flag 0: Not detect high-level 1: Detect high-level
0	INTRS	Rsing edge flag 0: Not detect rising edge 1: Detect rising edge

NAND flash/PC card common space timing configuration registers (EXMC_NPCTCFGx) (x=1, 2, 3)

Address offset: 0x48 + 0x20 * x, (x = 1, 2, and 3)
Reset value: 0xFFFFFFFF

These operations applicable to common memory space for 16-bit PC Card, CF card and 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 The bits are defined as time of bus keep high impedance state after writing the data. 0x00: COMHIZ = 1 * HCLK …… 0xFE: COMHIZ = 255 * HCLK 0xFF: Reserved
23:16	COMHLD[7:0]	Common memory hold time 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. 0x00: Reserved 0x01: COMHLD = 1 * HCLK …… 0xFE: COMHLD = 254 * HCLK 0xFF: Reserved
15:8	COMWAIT[7:0]	Common memory wait time Define the minimum time to maintain command 0x00: Reserved 0x01: COMWAIT = 2 * HCLK (+NWAIT active cycles) …… 0xFE: COMWAIT = 255 * HCLK (+NWAIT active cycles) 0xFF: Reserved
7:0	COMSET[7:0]	Common memory setup time Define the time to build address before sending command 0x00: COMSET = 1 * HCLK …… 0xFE: COMSET = 255 * HCLK 0xFF: Reserved

NAND flash/PC card attribute space timing configuration registers (EXMC_NPATCFGx) (x=1, 2, 3)

Address offset: 0x4C + 0x20 * x, (x = 1, 2, and 3)
Reset value: 0xFFFFFFFF

It is used for 8-bit accesses to the attribute memory space of the PC Card or to access 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 The bits are defined as time of bus keep high impedance state after writing the data. 0x00: ATTHIZ = 0 * HCLK …... 0xFE: ATTHIZ = 254 * HCLK 0xFF: Reserved
23:16	ATTHLD[7:0]	Attribute memory hold time 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. 0x00: Reserved 0x01: ATTHLD = 1 * HCLK …… 0xFE: ATTHLD = 254 * HCLK 0xFF: Reserved
15:8	ATTWAIT[7:0]	Attribute memory wait time Define the minimum time to maintain command 0x00: Reserved 0x01: ATTWAIT = 2 * HCLK (+NWAIT active cycles) …… 0xFE: ATTWAIT = 255 * HCLK (+NWAIT active cycles) 0xFF: ATTWAIT = Reserved
7:0	ATTSET[7:0]	Attribute memory setup time Define the time to build address before sending command 0x00: ATTSET = 1 * HCLK …… 0xFE: ATTSET = 255 * HCLK 0xFF: Reserved

PC card I/O space timing configuration register (EXMC_PIOTCFG3)

Address offset: 0xB0
Reset value: 0xFFFFFFFF

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IOHIZ[7:0]								IOHLD[7:0]							
rw								rw							

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

Bits	Fields	Descriptions
31:24	IOHIZ[7:0]	IO space data bus HiZ time The bits are defined as time of bus keep high impedance state after writing the data. 0x00: IOHIZ = 0 * HCLK …… 0xFF: IOHIZ = 255 * HCLK
23:16	IOHLD[7:0]	IO space hold time 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. 0x00: Reserved 0x01: IOHLD = 1 * HCLK …… 0xFF: IOHLD = 255 * HCLK
15:8	IOWAIT[7:0]	IO space wait time Define the minimum time to maintain command 0x00: Reserved 0x01: IOWAIT = 2 * HCLK (+NWAIT active cycles) …… 0xFF: IOWAIT = 256 * HCLK (+NWAIT active cycles)
7:0	IOSET[7:0]	IO space setup time Define the time to build address before sending command 0x00: IOSET = 1 * HCLK …… 0xFF: IOSET = 256 * HCLK

NAND flash ECC registers (EXMC_NECCx) (x=1, 2)

Address offset: 0x54+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
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 If ECCSZ[2:0] = 000, the page size of NAND Flash is 256 bytes, the ECC result is stored in ECC[21:0]. If ECCSZ[2:0] = 001, the page size of NAND Flash is 512 bytes, the ECC result is stored in ECC[23:0]. If ECCSZ[2:0] = 010, the page size of NAND Flash is 1024 bytes, the ECC result is stored in ECC[25:0]. If ECCSZ[2:0] = 011, the page size of NAND Flash is 2048 bytes, the ECC result is stored in ECC[27:0]. If ECCSZ[2:0] = 100, the page size of NAND Flash is 4096 bytes, the ECC result is stored in ECC[29:0]. If ECCSZ[2:0] = 101, the page size of NAND Flash is 8192 bytes, the ECC result is stored in ECC[31:0].

30.4.3. 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
Reserved	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	

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14:13	PIPED[1:0]	Pipeline delay These bits specify the delay for reading data after CAS latency in HCLK clock cycles. 00: 0 HCLK clock cycle delay 01: 1 HCLK clock cycle delay 10: 2 HCLK clock cycle delay 11: reserved **Note:** The corresponding bits in the EXMC_SDCTL1 register are reserved.

12	BRSTRD	Burst read When this bit is set, the SDRAM controller anticipates the next read commands during the CAS latency and stores data in the read FIFO. 0: Disable burst read 1: Enable burst read **Note:** The corresponding bits in the EXMC_SDCTL1 register are reserved.
11:10	SDCLK[1:0]	SDRAM clock configuration 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. 00: SDCLK memory clock disabled 01: Reserved 10: SDCLK memory period = 2 x HCLK periods 11: SDCLK memory period = 3 x HCLK periods **Note:** The corresponding bits in the EXMC_SDCTL1 register are reserved.
9	WPEN	Write protection enable This bit enables the write protection function. 0: Disable write protection, write accesses allowed 1: Enable write protection, write accesses ignored
8:7	CL[1:0]	CAS Latency This bits sets specifies SDRAM CAS latency in SDRAM memory clock cycle unit 00: reserved, do not use. 01: 1 cycle 10: 2 cycles 11: 3 cycles
6	NBK	Number of banks This bit specifies the number of internal banks. 0: 2 internal Banks 1: 4 internal Banks
5:4	SDW[1:0]	SDRAM data bus width. These bits specify the SDRAM memory data width. 00: 8 bits 01: 16 bits 10: 32 bits 11: reserved
3:2	RAW[1:0]	Row address bit width These bits specify the bit width of a row address. 00: 11 bit 01: 12 bits 10: 13 bits 11: reserved

1:0	CAW[1:0]	Column address bit width These bits specify the bit width of column address. 00: 8 bits 01: 9 bits 10: 10 bits 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	Must be kept at reset value.
27:24	RCD[3:0]	Row to column delay These bits specify the delay between an Activate command and a read/write command in SDRAM memory clock cycle unit. 0x0: 1 cycle. 0x1: 2 cycles .... 0xF: 16 cycles
23:20	RPD[3:0]	Row precharge delay These bits specify the delay between a precharge command and the next command in SDRAM memory clock cycle unit. 0x0: 1 cycle 0x1: 2 cycles .... 0xF: 16 cycles **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 These bits specify the delay between a write and a precharge command in SDRAM memory clock cycle unit. 0x0: 1 cycle

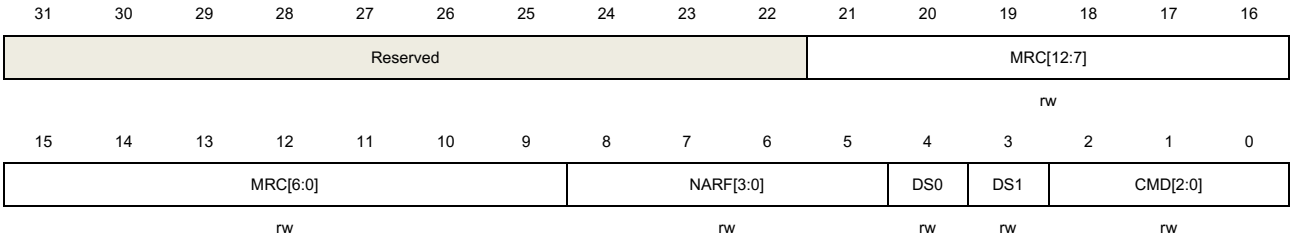
Bits	Fields	Descriptions
		0x1: 2 cycles ..... 0xF: 16 cycles **Note:** The corresponding bits in the EXMC_SDTCFG1 register are reserved. If two SDRAM memories are used, the WRD must be programmed with the timings of the slower one.
15:12	ARFD[3:0]	Auto refresh delay These bits specify the delay between 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. 0x0: 1 cycle 0x1: 2 cycles .... 0xF: 16 cycles **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 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. 0x0: 1 cycle 0x1: 2 cycles ..... 0xF: 16 cycles
7:4	XSRD[3:0]	Exit self-refresh delay These bits specify the delay from a self-refresh command to an activate command in SDRAM memory clock cycle unit. 0x0: 1 cycle 0x1: 2 cycles …… 0xF: 16 cycles
3:0	LMRD[3:0]	Load mode register delay These bits specify the delay between a load mode register command and a refresh or active command in SDRAM memory clock cycle unit. 0x0: 1 cycle 0x1: 2 cycles ..... 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)



Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21:9	MRC[12:0]	Mode register content 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'. 0x0: 1 Auto-refresh cycle 0x1: 2 Auto-refresh cycles .... 0xE: 15 Auto-refresh cycles 0xF: Reserved
4	DS0	Device select 0 This bit indicates whether the SDRAM Device0 is selected or not. 0: SDRAM Device0 is not selected 1: SDRAM Device0 is selected
3	DS1	Device select 1 This bit indicates whether the SDRAM Device1 is selected or not. 0: SDRAM Device1 is not selected 1: SDRAM Device1 is selected
2:0	CMD[2:0]	Command These bits specify the commands, which are issued to the SDRAM device. 000: Normal operation command 001: Clock enable command 010: Precharge all command 011: Auto-refresh command 100: Load mode register command 101: Self-refresh command

110: Power-down entry command

111: Reserved

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)

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	REIE	ARINTV[12:0]													REC
	rw	rw													w

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	REIE	Refresh error interrupt enable 0: Disable 1: Enable. An interrupt is generated if REIF bit of the status register is set
13:1	ARINTV[12:0]	Auto-refresh interval This bit field specifies the interval of two successive auto-refresh commands in memory clock cycle unit. ARFITV = (SDRAM refresh period / Number of rows) - 20
0	REC	Refresh error flag clear The refresh error flag (REIF) in the status register will be cleared when this bit is set. 0: no effect 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)

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	NRDY	STA1[1:0]	STA0[1:0]	REIF
	r	r	r	r

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	NRDY	Not ready status This bit specifies whether the SDRAM controller is ready for a new command 0: SDRAM controller is ready for a new command 1: SDRAM controller is not ready for a new command
4:3	STA1[1:0]	Device1 status This bit defines the Status of SDRAM Device1. 00: Normal status 01: Self-refresh status 10: Power-down status
2:1	STA0[1:0]	Device 0 status This bit defines the Status of SDRAM Device 0. 00: Normal status 01: Self-refresh status 10: Power-down status
0	REIF	Refresh error interrupt flag 0: No refresh error 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	Must be kept at reset value.
7:4	SDSC[3:0]	Select the delayed sample clock of read data 0x0: Select the clock after 0 delay cell 0x1: Select the clock after 1 delay cell ……

		0xF: Select the clock after 15 delay cell
3:2	Reserved	Must be kept at reset value.
1	SSCR	Select sample cycle of read data 0: add 0 extra HCLK cycle to the read data sample clock besides the delay chain 1: add 1 extra HCLK cycle to the read data sample clock besides the delay chain
0	RSEN	Read sample enable 0: Disable read sample 1: Enable read sample

30.4.4. SQPI-PSRAM controller registers

SPI initialization register(EXMC_SINIT)

Address offset: 0x310

Reset value: 0x1801 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL	IDL[1:0]		ADRBIT[4:0]					Reserved						CMDBIT[1:0]	
rw	rw		rw											rw	

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

Bits	Fields	Descriptions
31	POL	Read data sample polarity. 0: Sample data at rising edge(default) 1: Sample data at falling edge.
30:29	IDL[1:0]	SPI PSRAM ID Length. 00:64-bit 01:32-bit 10:16-bit 11:8-bit
28:24	ADRBIT[4:0]	Bit number of SPI PSRAM address phase. Value Range:1 to 26(default:24) 0x00: reserved 0x01: 1-bit address …… 0x1A: 26-bit address 0x1B: reserved ……

		0x1F: reserved
23:18	Reserved	Must be kept at reset value.
17:16	CMDBIT[1:0]	Bit number of SPI PSRAM command phase 00: 4 bit 01: 8 bit (default) 10: 16 bit 11: Reserved
15:0	Reserved	Must be kept at reset value.

SPI read command register(EXMC_SRCMD)

Address offset: 0x320

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDID	Reserved									RMODE[1:0]		RWAITCYCLE[3:0]			
rw										rw		rw			

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

Bits	Fields	Descriptions
31	RDID	Send SPI Read ID Command, command code and mode come from RCMD and RMODE.
30:22	Reserved	Must be kept at reset value.
21:20	RMODE[1:0]	SPI PSRAM read command mode 00: Not SPI mode 01: SPI mode 10: SQPI mode 11: QPI mode
19:16	RWAITCYCLE[3:0]	SPI read wait cycle number after address phase.
15:0	RCMD[15:0]	SPI read command for AHB read transfer. When CMDBIT is different, valid RCMD is different: CMDBIT=00,RCMD[3:0] are valid. CMDBIT=01,RCMD[7:0] are valid. CMDBIT=10,RCMD[15:0] are valid.

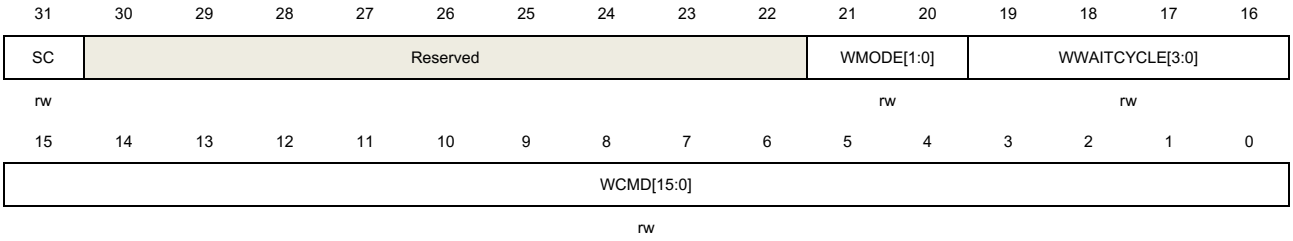
Note: Before writing 1 to RDID bit, users must ensure it is cleared by reading RDID as 0.

SPI write command register(EXMC_SWCMD)

Address offset: 0x330

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	SC	Send SPI special command which does not have address and data phase, command code and mode come from WCMD and WMODE.
30:22	Reserved	Must be kept at reset value.
21:20	WMODE[1:0]	SPI PSRAM write command mode 00: Not SPI mode 01: SPI mode 10: SQPI mode 11: QPI mode
19:16	WWAITCYCLE[3:0]	SPI write wait cycle number after address phase
15:0	WCMD[15:0]	SPI write command for AHB write transfer

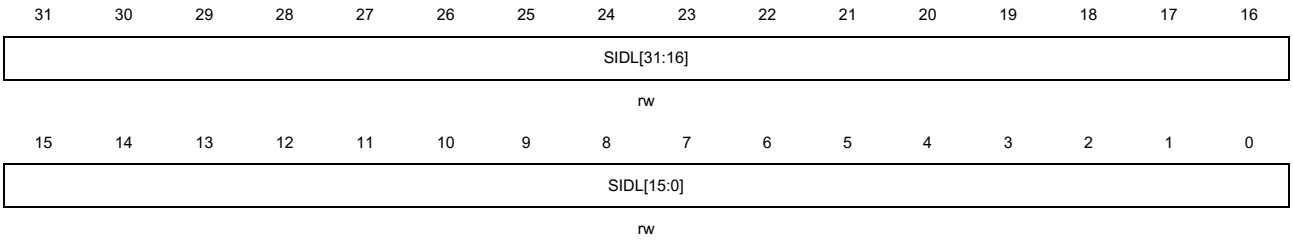
Note: Before write 1 to SC bit, you must ensure it is cleared and after set SC to 1, you must wait SC cleared.

SPI ID low register(EXMC_SIDL)

Address offset: 0x340

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:0	SIDL[31:0]	ID low data saved for SPI read ID command SIDL[31:0] is valid when IDL=01 or 00.

SIDL[15:0] is valid when IDL=10.

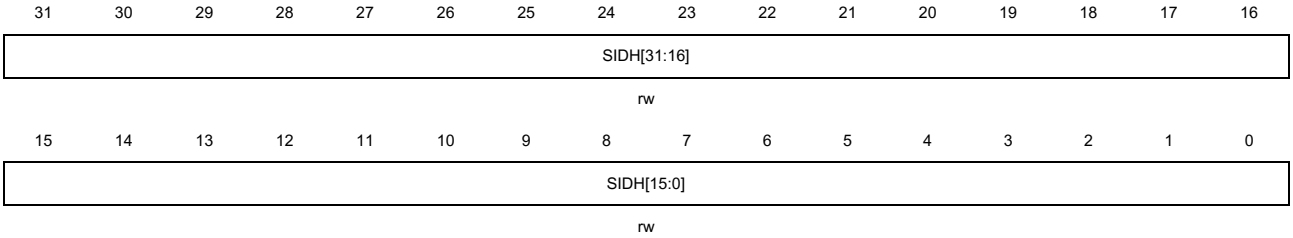
SIDL[7:0] is valid when IDL=11.

SPI ID high register(EXMC_SIDH)

Address offset: 0x350

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:0	SIDH[63:32]	ID high data saved for SPI read ID command. **Note:** SIDH[31:0] is valid when IDL=00.

31. Controller area network (CAN)

31.1. 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.

As CAN network interface, basic extended CAN supports the CAN protocols version 2.0A, 2.0B, ISO11898-1:2015 and BOSCH CAN FD specification. The CAN interface automatically handles the transmission and the reception of CAN frames. The CAN provides 28 scalable/configurable identifier filter banks. The filters are used for selecting the input message as software requirement and otherwise discarding the message. Three transmit mailboxes are provided to the software for transfer messages. The transmission scheduler decides which mailbox will be transmitted firstly. Three complete messages can be stored in every FIFO. The FIFOs are managed completely by hardware. Two receiving FIFOs are used by hardware to store the incoming messages. In addition, the CAN controller provides all hardware functions, which supports the time-triggered communication option, in safety-critical applications.

31.2. Characteristics

- Supports CAN protocols version 2.0A, B.
- Supports CAN FD Frame with up to 64 data bytes (ISO11898-1 and Bosch CAN FD specification V1.0).
- Baud rates up to 1 Mbit/s when classical frames and 6 Mbit/s when FD frames.
- Supports transmitter delay compensation.
- Supports the time-triggered communication.
- Interrupt enable and clear.

Transmission

- Supports 3 transmit mailboxes.
- Supports priority of transmission message.
- Supports time stamp at SOF transmission.

Reception

- Supports 2 Rx FIFOs and each has 3 messages depth.
- 28 scalable/configurable identifier filter banks.
- FIFO lock.

Time-triggered communication

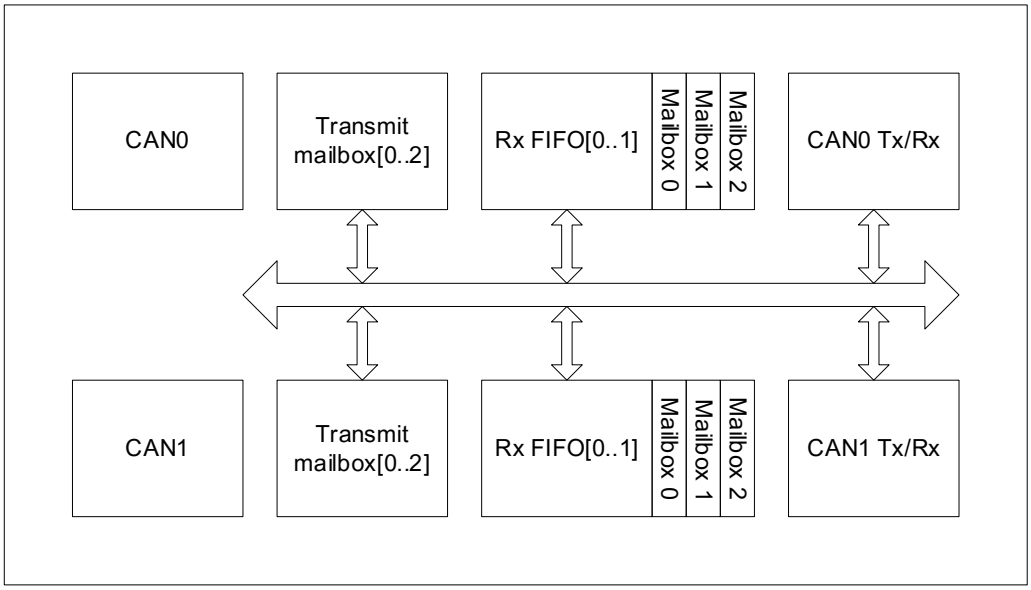
- Disable retransmission automatically in time-triggered communication mode.
- 16-bit free timer.

- Time stamp on SOF reception.
- Time stamp in last two data bytes transmission.

31.3. Function overview

Figure 31-1. CAN module block diagram shows the CAN block diagram.

Figure 31-1. CAN module block diagram



31.3.1. Working mode

The CAN interface has three working modes:

- Sleep working mode.
- Initial working mode.
- Normal working mode.

Sleep working mode

Sleep working mode is the default mode after reset. In sleep working mode, the CAN is in the low-power status and the CAN clock is stopped.

When SLPWMOD bit in CAN_CTL register is set, the CAN enters the sleep working mode. Then the SLPWS bit in CAN_STAT register is set by hardware.

To leave sleep working mode automatically: the AWU bit in CAN_CTL register is set and the CAN bus activity is detected. To leave sleep working mode by software: clear the SLPWMOD bit in CAN_CTL register.

Sleep working mode to initial working mode: set IWMOD bit and clear SLPWMOD bit in CAN_CTL register.

Sleep working mode to normal working mode: clear IWMOD and SLPWMOD bit in CAN_CTL register.

Initial working mode

When the configuration of CAN bus communication is needed to be changed, the CAN must enter initial working mode.

When IWMOD bit in CAN_CTL register is set, the CAN enters the initial working mode. Then the IWS bit in CAN_STAT register is set.

Initial working mode to sleep working mode: set SLPWMOD bit and clear IWMOD bit in CAN_CTL register.

Initial working mode to normal working mode: clear IWMOD bit and clear SLPWMOD bit in CAN_CTL register.

Normal working mode

The CAN could communicate with other CAN communication nodes in normal working mode

To enter normal working mode: clear IWMOD and SLPWMOD bit in CAN_ CTL register.

Normal working mode to sleep working mode: set SLPWMOD bit in CAN_CTL register and wait the current transmission or reception completed.

Normal working mode to initial working mode: set IWMOD bit in CAN_CTL register, and wait the current transmission or reception completed.

31.3.2. Communication modes

The CAN interface has four communication modes:

- Silent communication mode.
- Loopback communication mode.
- Loopback and silent communication mode.
- Normal communication mode.

Silent communication mode

Silent communication mode means reception available and transmission disable.

The RX pin of the CAN could detect the signal from the network and the TX pin always holds in recessive state.

When the SCMOD bit in CAN_BT register is set, the CAN enters the silent communication mode. When it is cleared, the CAN leaves silent communication mode.

Silent communication mode is useful for monitoring the network messages.

Loopback communication mode

Loopback communication mode means the transmitted messages are transferred into the Rx FIFOs, the RX pin is disconnected from the CAN network and the TX pin can still send messages to the CAN network.

Setting LCMOD bit in CAN_BT register to enter loopback communication mode, while clearing it to leave. Loopback communication mode is useful for self-test.

Loopback and silent communication mode

Loopback and silent communication mode means the RX and TX pins are disconnected from the CAN network while the transmitted messages are transferred into the Rx FIFOs.

Setting LCMOD and SCMOD bit in CAN_BT register to enter loopback and silent communication mode, while clearing them to leave.

Loopback and silent communication mode is used for self-test. The TX pin holds in recessive state. The RX pin holds in high impedance state.

Normal communication mode

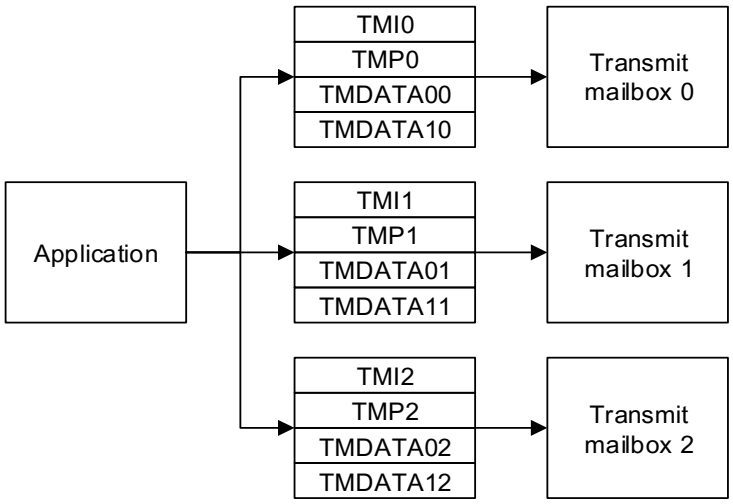
Normal communication mode is the default communication mode when the LCMOD and SCMOD bits in CAN_BT register are cleared.

31.3.3. Data transmission

Transmission register

Three transmit mailboxes are used for the application. Transmit mailboxes are used by configuring four transmission registers: CAN_TMIx, CAN_TMPx, CAN_TMDATA0x and CAN_TMDATA1x. As is shown in *Figure 31-2. Transmission register*.

Figure 31-2. Transmission register

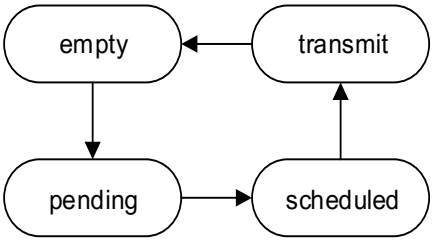


If FD frame would be transmitted, always write TMDATA00 registers when mailbox 0 is used, TMDATA01 register when mailbox 1 is used and TMDATA02 register when mailbox 2 is used until the end. For example, if application wants to transmit 64 bytes data using mailbox0, it needs to write the 64 bytes data through TMDATA00 register for 16 times. The data is stored in internal SRAM.

Transmit mailbox state

A transmit mailbox can be used when it is free (**empty state**). If the mailbox is filled with data, set TEN bit in CAN_TMIx register to prepare for starting the transmission **(pending state**). If more than one mailbox is in the pending state, they need scheduling the transmission (**scheduled state**). A mailbox with highest priority enters into transmit state and starts transmitting the message (**transmit state**). After the message has been sent, the mailbox is free (**empty state**). As is shown in *Figure 31-3. State of transmit mailbox*.

Figure 31-3. State of transmit mailbox



Transmit status and error

The CAN_TSTAT register includes the transmit status and error bits: MTF, MTFNERR, MAL, MTE.

- MTF: mailbox transmit finished. Typically, MTF is set when the frame in the transmit mailbox has been sent.
- MTFNERR: mailbox transmit finished with no error. MTFNERR is set when the frame in the transmit mailbox has been sent without any error.
- MAL: mailbox arbitration lost. MAL is set when the frame transmission is failed due to the arbitration lost.
- MTE: mailbox transmit error. MTE is set when the frame transmission is failed due to the error detected on the CAN bus.

Steps of sending a frame

To send a frame through the CAN:

Step 1: Select one free transmit mailbox.

Step 2: Configure four transmission registers with the application's acquirement.

Step 3: Set TEN bit in CAN_TMIx register.

Step 4: Check the transmit status. Typically, MTF and MTFNERR are set if transmission is

successful.

Transmission options

Abort

MST bit in CAN_TSTAT register can abort the transmission.

If the transmit mailbox's status is **pending** or **scheduled**, the abort of transmission can be done immediately.

In the **transmit** state, the abort of transmission does not take effect immediately until the transmission is finished. In case that the transmission is successful, the MTFNERR and MTF in CAN_TSTAT are set and state changes to be **empty**. In case that the transmission is failed, the state changes to be **scheduled** and then the abort of transmission can be done immediately.

Priority

When more than one transmit mailbox is pending, the transmission order is given by the TFO bit in CAN_CTL register.

In case that TFO is 1, the three transmit mailboxes work first-in first-out (FIFO).

In case that TFO is 0, the transmit mailbox with lowest identifier has the highest priority of transmission. If the identifiers are equal, the lower mailbox number will be scheduled firstly.

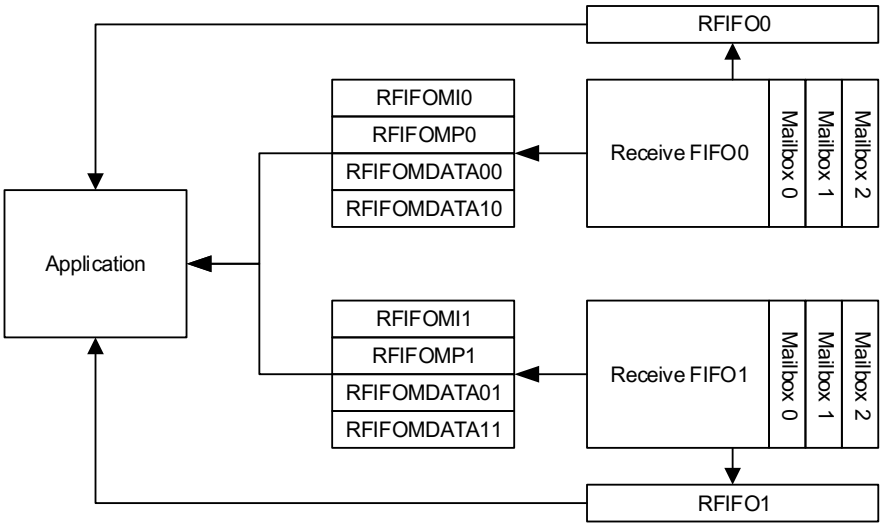
31.3.4. Data reception

Reception register

Two Rx FIFOs are used for the application. Rx FIFOs are managed by five registers: CAN_RFIFOx, CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x. FIFO's status and operation can be handled by CAN_RFIFOx register. Reception frame data can be achieved through the registers: CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x.

Each FIFO consists of three receive mailboxes. As is shown in ***Figure 31-4. Reception register***.

Figure 31-4. Reception register



Rx FIFO

Rx FIFO has three mailboxes. The reception frames are stored in the mailbox according to the arriving sequence. First arrived frame can be accessed by application firstly.

The number of frames in the Rx FIFO and the status can be accessed by the register CAN_RFIFO0 and CAN_RFIFO1.

If at least one frame has been stored in the Rx FIFO0, the frame data is stored in the CAN_RFIFOMI0, CAN_RFIFOMP0, CAN_RFIFOMDATA00 and CAN_RFIFOMDATA10 registers. After reading the current frame, set RFD bit in CAN_RFIFO0 to release a frame in the Rx FIFO and the software can read the next frame.

If FD frame has been received, the data always read from CAN_RFIFOMDATA00 register for FIFO0 and CAN_RFIFOMDATA01 for FIFO1 until the end. For example, if application needs to read 64 bytes data from FIFO0. It needs to read the 64 bytes data through CAN_RFIFOMDATA00 register for 16 times. The received data is stored in internal SRAM.

Rx FIFO status

RFL (Rx FIFO length) bits in CAN_RFIFOx register is 0 when no frame is stored in the Rx FIFO and it is 3 when FIFOx is full.

When RFF bit in CAN_RFIFOx register is set, it indicates FIFOx is full, at this time, RFL is 3.

When a new frame arrives after the FIFO has held three frames, the RFO bit in CAN_RFIFOx register will be set, and it indicates FIFOx is overrun. If the RFOD bit in CAN_CTL register is set, the new frame is discarded. If the RFOD bit in CAN_CTL register is reset, the new frame is stored into the Rx FIFO and the last frame in the Rx FIFO is discarded.

Steps of receiving a message

Step 1: Check the number of frames in the Rx FIFO.

Step 2: Read CAN_RFIFOMIx, CAN_RFIFOMPx, CAN_RFIFOMDATA0x and CAN_RFIFOMDATA1x.

Step 3: Set the RFD bit in CAN_RFIFOx register.

31.3.5. Filtering function

The CAN receives frames from the CAN bus. If the frame passes the filter, it is stored in the Rx FIFOs. Otherwise, the frame will be discarded without intervention by the software.

The identifier of frame is used for the matching of the filter.

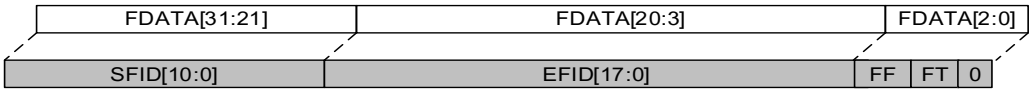
Scale

The filter consists of 28 banks: bank0 to bank27. Each bank has two 32-bit registers: CAN_FxDATA0 and CAN_FxDATA1.

Each filter bank can be configured to 32-bit or 16-bit.

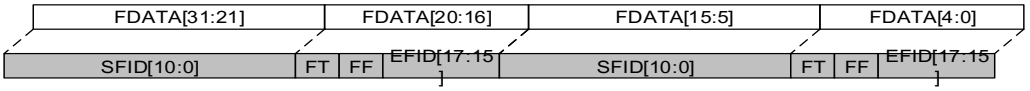
32-bit: SFID[10:0], EFID[17:0], FF and FT bits. As is shown in *Figure 31-5. 32-bit filter*.

Figure 31-5. 32-bit filter



16-bit: SFID[10:0], FT, FF and EFID[17:15] bits. As is shown in *Figure 31-6. 16-bit filter*.

Figure 31-6. 16-bit filter



Mask mode

For the Identifier of a data frame to be filtered, the mask mode is used to specify which bits must be the same as the preset Identifier and which bits need not be judged. 32-bit mask mode example is shown in *Figure 31-7. 32-bit mask mode filter*.

Figure 31-7. 32-bit mask mode filter

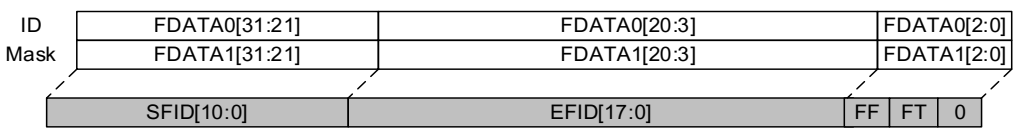
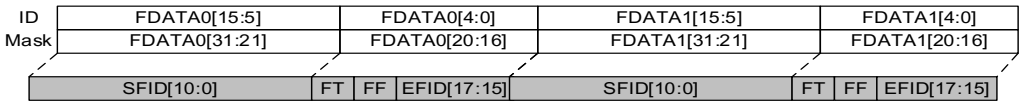


Figure 31-8. 16-bit mask mode filter



List mode

The filter consists of frame identifiers. The filter can determine whether a frame will be discarded or not. When one frame arrived, the filter will check which member can match the identifier of the frame.

32-bit list mode example is shown in ***Figure 31-9. 32-bit list mode filter***.

Figure 31-9. 32-bit list mode filter

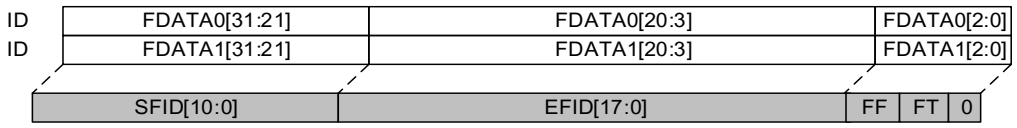
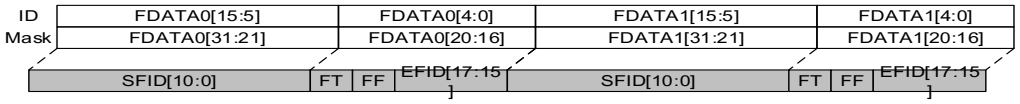


Figure 31-10. 16-bit list mode filter



Filter number

Filter consists of some filter bank. According to the mode and the scale of each of the filter banks, filter has different effects.

For example, there are two filter banks. Bank0 is configured as 32-bit mask mode. Bank1 is configured as 32-bit list mode. The filter number is shown in ***Table 31-1. 32-bit filter number***.

Table 31-1. 32-bit filter number

Filter bank	Filter data register	Filter number
0	F0DATA0-32bit-ID	0
	F0DATA1-32bit-Mask	
1	F1DATA0-32bit-ID	1
	F1DATA1-32bit-ID	2

Associated FIFO

28 banks can be associated with FIFO0 or FIFO1. If the bank is associated with FIFO0, the frames passed the bank will be stored in the FIFO0.

Active

The filter bank needs to be activated if the bank is to be used, otherwise, the filter bank should

be left deactivated.

Filtering index

Each filter number corresponds to a filtering rule. When the frame which is associated with a filter number N passes the filters, the filter index is N. It stores in the FI bits in CAN_RFIFOMPx.

Filter bank has filter index once it is associated with the FIFO no matter whether the bank is active or not.

The example about filtering index is shown in *Table 31-2. Filtering index*.

Table 31-2. Filtering index

Filter bank	FIFO0	Active	Filter nunber	Filter bank	FIFO1	Active	Filter nunber
0	F0DATA0-32bits-ID	Yes	0	2	F2DATA0[15:0]-16bits-ID	Yes	0
	F0DATA1-32bits-Mask				F2DATA0[31:16]-16bits-Mask		
1	F1DATA0-32bits-ID	Yes	1		F2DATA1[15:0]-16bits-ID		1
	F1DATA1-32bits-ID		2		F2DATA1[31:16]-16bits-Mask		
3	F3DATA0[15:0]-16bits-ID	No	3	4	F4DATA0-32bits-ID	No	2
	F3DATA0[31:16]-16bits-Mask				F4DATA1-32bits-Mask		
	F3DATA1[15:0]-16bits-ID		4	5	F5DATA0-32bits-ID	No	3
	F3DATA1[31:16]-16bits-Mask				F5DATA1-32bits-ID		4
7	F7DATA0[15:0]-16bits-ID	No	5	6	F6DATA0[15:0]-16bits-ID	Yes	5
	F7DATA0[31:16]-16bits- ID		6		F6DATA0[31:16]-16bits- ID		6
	F7DATA1[15:0]-16bits-ID		7		F6DATA1[15:0]-16bits-ID		7
	F7DATA1[31:16]-16bits- ID		8		F6DATA1[31:16]-16bits- ID		8
8	F8DATA0[15:0]-16bits-ID	Yes	9	10	F10DATA0[15:0]-16bits-ID	No	9
	F8DATA0[31:16]-16bits- ID		10		F10DATA0[31:16]-16bits-Mask		
	F8DATA1[15:0]-16bits-ID		11		F10DATA1[15:0]-16bits-ID		10
	F8DATA1[31:16]-		12		F10DATA1[31:16]-		

Filter bank	FIFO0	Active	Filter nunber	Filter bank	FIFO1	Active	Filter nunber
	16bits- ID				16bits-Mask		
9	F9DATA0[15:0]-16bits-ID	Yes	13	11	F11DATA0[15:0]-16bits-ID	No	11
	F9DATA0[31:16]-16bits-Mask				F11DATA0[31:16]-16bits- ID		12
	F9DATA1[15:0]-16bits-ID		14		F11DATA1[15:0]-16bits-ID		13
	F9DATA1[31:16]-16bits-Mask				F11DATA1[31:16]-16bits- ID		14
12	F12DATA0-32bits-ID	Yes	15	13	F13DATA0-32bits-ID	Yes	15
	F12DATA1-32bits-Mask				F13DATA1-32bits-ID		16

Priority

The filters have the priority rules:

1. 32-bits mode is higher than 16-bits mode.
2. List mode is higher than mask mode.
3. Smaller filter number has the higher priority.

31.3.6. Time-triggered communication

The time-triggered CAN protocol is a higher layer protocol on top of the CAN data link layer. Time-triggered communication means that activities are triggered by the elapsing of time segments. In a time-triggered communication system, all time points of message transmission are pre-defined.

In this mode, an internal 16-bit counter starts working, incrementing by 1 at each CAN bit time. This internal counter provides time stamps for sending and receiving data, stored in registers CAN _RFIFOMPx and CAN_TMPx.

The automatic retransmission is disabled in the time-triggered CAN communication.

31.3.7. Communication parameters

Automatic retransmission forbid mode

In time-triggered communication mode, the requirement for automatic retransmission must be disabled and CAN be met by setting ARD position 1 of the CAN_CTL register.

In this mode, the data is sent only once, and if the transmission fails due to arbitration failure

or bus error, the CAN bus controller does not automatically resend the data as usual.

At the end of sending, the MTF bit of register CAN_TSTAT is hardware set to 1, and the sending status information can be obtained via MTFNERR, MAL, and MTE.

Bit time

On the bit-level, the CAN protocol uses synchronous bit transmission. This not only enhances the transmitting capacity but also requires a sophisticated method of bit synchronization. While bit synchronization in a character-oriented transmission (asynchronous) is performed upon the reception which the start bit is available with each character, the synchronous transmission protocol just need one start bit available at the beginning of a frame. To ensure that the receiver correctly reads the messages, resynchronization is required. Phase buffer segments' sample point of the front-end and back-end should be inserted a bit interval.

The CAN protocol regulates bus access by bit-wise arbitration. The signal propagated from sender to receiver and back to the sender must be completed within one bit-time. For synchronization, in addition to the phase buffer segments, a propagation delay segment is needed. The propagation delay segment is regarded as signal delays caused by transmitting and receiving nodes in the process of the signal propagation on the bus.

The normal bit time from the CAN protocol has three segments as follows:

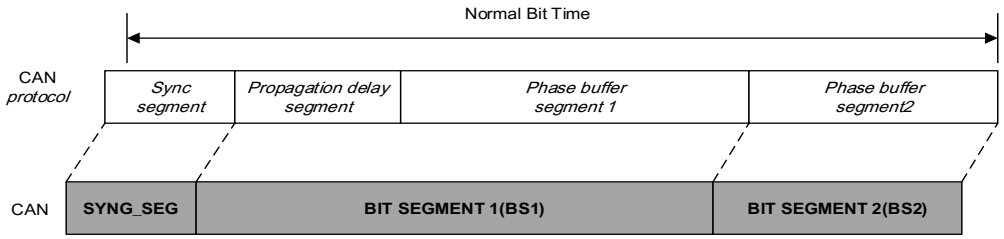
Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).

Bit segment 1 (BS1): It defines the location of the sample point. It includes the Propagation delay 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 transmit point. It represents the Phase buffer segment 2 in the CAN standard. It may also be automatically shortened to compensate for negative phase drifts.

The bit time is shown as in the ***Figure 31-11. The bit time***.

Figure 31-11. The bit time



The resynchronization Jump Width (SJW): it can be lengthened or shortened to compensate for the Synchronization error of the CAN network node.

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 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 to SJW at most, so that the transmit point is moved earlier.

Note: For detailed explanations of hard synchronization and resynchronization, please refer to the ISO 11898 standard.

Baud rate

The clock of the CAN derives from the APB1 bus. The CAN calculates its baud rate as follow:

$$BaudRate=\frac{1}{Normal\ Bit\ Time} \quad (31\text{-}1)$$

$$Normal\ Bit\ Time=t_{SYNC_SEG}+t_{BS1}+t_{BS2} \quad (31\text{-}2)$$

with:

$$t_{SYNC_SEG}=1\times t_q \quad (31\text{-}3)$$

$$t_{BS1}=(1+BT.BS1)\times t_q \quad (31\text{-}4)$$

$$t_{BS2}=(1+BT.BS2)\times t_q \quad (31\text{-}5)$$

$$t_q=(1+BT.BAUDPSC)\times t_{PCLK1} \quad (31\text{-}6)$$

31.3.8. CAN FD operation

The CAN FD function is enabled by setting FDEN to 1 in CAN_FDCTL register. If FDEN bit is cleared, only calassical frames are supported. If FDEN bit is set, it supports calassical frames and FD frames. Whether the current frame is FD or not could be defined by received FDF bit (the previously reserved bit in CAN frames with 11-bit identifiers or the first previously reserved bit in CAN frames with 29-bit identifiers which now is decoded as FDF bit). If FDF bit is recessive, meaning to be the CAN FD frame, otherwise FDF bit is dominant, meaning to be the classical frame.

The CAN FD supports ISO11898-1 or Bosch CAN FD Specification V1.0 by configuring NISO bit in CAN_FDCTL register.

The two bits following the FDF bit are reserved bit and BRS bit respectively. The received BRS bit determines the bit rate of data. When BRS is dominant, bit rate of data could not switch by configuring the CAN_DBT register. When BRS is recessive, bit rate of data could switch to a higher bit rate inside the frame by configuring the CAN_DBT register. The bit rate of data can be switched in the period from BRS bit to CRC delimiter (refer to ISO11898-1 or Bosch CAN FD Specification V1.0).

When Protocol Exception Handling is enabled (PRED bit in CAN_FDCTL register is 0), it causes the operation state to change to be IDLE and interrupts current frame at the next sample point when recessive reserve bit is received. When Protocol Exception Handling is disabled (PRED bit in CAN_FDCTL register is 1), it will treat a recessive reserve bit as a form error and respond with an error frame. If any recessive reserve bit occurs, set PRE bit in CAN_FDSTAT register to 1.

The transmission of ESI bit (the bit before DLC bits, refer to ISO11898-1 or Bosch CAN FD Specification V1.0) is defined by ESIMOD bit in CAN_FDCTL register and ESI bit in CAN_TMPx register. If ESIMOD bit is 0, it will transmit the dominant bit by error active nodes and transmit the recessive bit by error passive nodes. If ESIMOD bit is set, it will transmit ESI bit in CAN_TMPx register.

The transmission of FDF bit and BRS bit is defined by FDF bit and BRS bit in CAN_TMPx registers.

31.3.9. Transmitter Delay Compensation

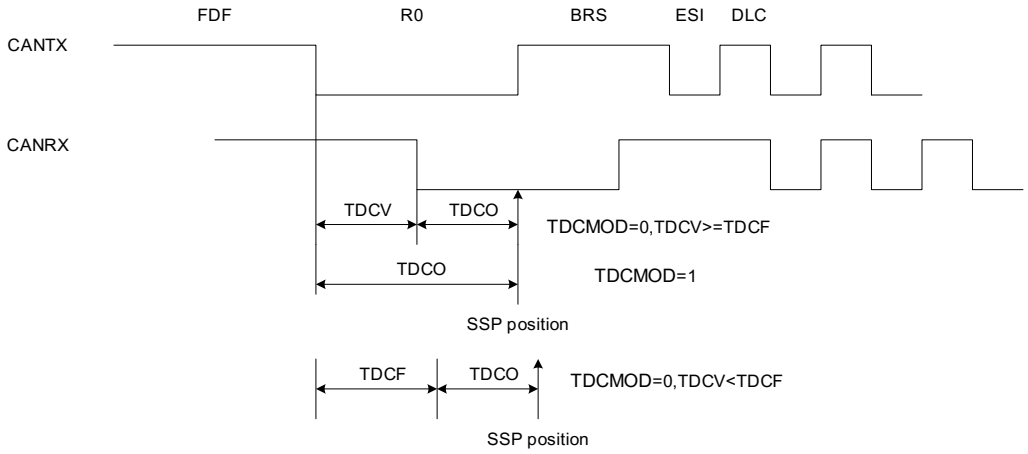
CAN FD supports transmitter delay compensation mechanism, it could be used in applications when the length of the CAN bit time in the DATA-PHASE is shorter than the limit required by the transceiver's internal delay time. The description of transmitter delay compensation, please refer to ISO11898-1 or Bosch CAN FD Specification V1.0.

The transmitter delay compensation is enabled by setting TDCEN bit in CAN_FDCTL register.

The transmitter delay compensation is used to adjust the position of the SSP. The SSP delay is defined as the delay from dominant edges on CANTX to SSP point. If TDCMOD bit in CAN_FDCTL register is set, the SSP delay is defined in TDCO bits of the CAN_FDTDC registers by software. If TDCMOD bit in CAN_FDCTL register is cleared, the hardware automatically uses the falling edges between FDF bit and RES0 bit to calculate the delay from dominant edges on CANTX to dominant edges on CANRX, and store the delay in TDCV bits of CAN_FDSTAT registers. In case that there is dominant glitch, SSP position would be advanced than expected, leading to a calculated error in compensation measurement. To solve this problem, TDCF bits of CAN_FDTDC register could be used to avoid too small TDCV bits. So the value of SSP delay is TDCO bits add TDCV bits if TDCV is larger than TDCF, or else the value of SSP delay is TDCO bits add TDCF bits.

The SSP delay can not exceed 3 data bit time.

Figure 31-12. Transmitter Delay Measurement



31.3.10. Error flags

The state of CAN bus can be reflected by Transmit Error Counter (TECNT) and Receive Error Counter (RECNT) of CAN_ERR register. The value can be increased or decreased by the hardware according to the error, and the software can judge the stability of the CAN network by these values. For details on incorrect counting, refer to the CAN protocol section.

Their values which read by software determine whether the CAN network is stable.By using the CAN_INTEN register (ERRIE bit, etc.), the software can control the interrupt generation when error is detected.

Bus-Off recovery

The CAN controller is in Bus-Off state when TECNT is over than 255. In This state, BOERR bit is set in CAN_ERR register, and no longer able to transmit and receive messages.

According to the ABOR configuration in register CAN_CTL, there are two ways to recover from Bus-Off (to an Error active state). Both of these methods require the CAN bus controller in the offline state to detect the Bus-Off recovery sequence defined by CAN protocol (when CAN_RX detects 128 consecutive 11-bit recessive bits) before automatic recovery.

If ABOR is set, it will be automatically recovered when a Bus-Off recovery sequence is detected.

If ABOR is cleared, CAN controller must be configured to enter initialization mode by setting IWMOD bit in CAN_CTL register, then exit and enter nomal mode. After this operation, it will recover when the recovering sequence is detected.

31.3.11. CAN interrupts

The CAN bus controller occupies 4 interrupt vectors, which are controlled by the register CAN _INTEN.

The interrupt sources can be classified as:

- Transmit interrupt.
- FIFO0 interrupt.
- FIFO1 interrupt.
- Error and status change interrupt.

Transmit interrupt

The transmit interrupt can be generated by any of the following conditions and TMEIE bit in CAN_INTEN register will be set:

- TX mailbox 0 transmit finished: MTF0 bit in the CAN_TSTAT register is set.
- TX mailbox 1 transmit finished: MTF1 bit in the CAN_TSTAT register is set.
- TX mailbox 2 transmit finished: MTF2 bit in the CAN_TSTAT register is set.

Rx FIFO0 interrupt

The Rx FIFO0 interrupt can be generated by the following conditions:

- Rx FIFO0 not empty: RFL0 bits in the CAN_RFIFO0 register are not '00' and RFNEIE0 in CAN_INTEN register is set.
- Rx FIFO0 full: RFF0 bit in the CAN_RFIFO0 register is set and RFFIE0 in CAN_INTEN register is set.
- Rx FIFO0 overrun: RFO0 bit in the CAN_RFIFO0 register is set and RFOIE0 in CAN_INTEN register is set.

Rx FIFO1 interrupt

The Rx FIFO1 interrupt can be generated by the following conditions:

- Rx FIFO1 not empty: RFL1 bits in the CAN_RFIFO1 register are not '00' and RFNEIE1 in CAN_INTEN register is set.
- Rx FIFO1 full: RFF1 bit in the CAN_RFIFO1 register is set and RFFIE1 in CAN_INTEN register is set.
- Rx FIFO1 overrun: RFO1 bit in the CAN_RFIFO1 register is set and RFOIE1 in CAN_INTEN register is set.

Error and working mode change interrupt

The error and working mode change interrupt can be generated by the following conditions:

- Error: ERRIF bit in the CAN_STAT register and ERRIE bit in the CAN_INTEN register are set. Refer to ERRIF description in the CAN_STAT register.
- Wakeup: WUIF bit in the CAN_STAT register is set and WIE bit in the CAN_INTEN register is set.
- Enter sleep working mode: SLPIF bit in the CAN_STAT register is set and SLPWIE bit in the CAN_INTEN register is set.

The CAN bus controller interrupt conditions can refer to ***Table 31-3. CAN Event / Interrupt flags***.

Table 31-3. CAN Event / Interrupt flags

Interrupt event	Interrupt / Event flag		Enable control bit	
Transmit interrupt	Mailbox 0 transmit finished flag (MTF0)		TMEIE	
	Mailbox 1 transmit finished flag (MTF1)			
	Mailbox 2 transmit finished flag (MTF2)			
FIFO0 interrupt	Rx FIFO0 length (RFL0[1:0])		RFNEIE0	
	Rx FIFO0 full (RFF0)		RFFIE0	
	Rx FIFO0 overfull (RFO0)		RFOIE0	
FIFO1 interrupt	Rx FIFO1 length (RFL0[1:0])		RFNEIE1	
	Rx FIFO1 full (RFF0)		RFFIE1	
	Rx FIFO1 overfull (RFO0)		RFOIE1	
EWMC interrupt	Warning error (WERR)	Error interrupt flag (ERRIF)	WERRIE	ERRIE
	Passive error (PERR)		PERRIE	
	Bus-Off error (BOERR)		BOIE	
	Error number (1<= ERRN[2:0] <= 6)		ERRNIE	
	Status change interrupt flag of waking up from sleep working mode (WUIF)		WIE	
	Status change interrupt flag of entering sleep working mode (SLPIF)		SLPWIE	

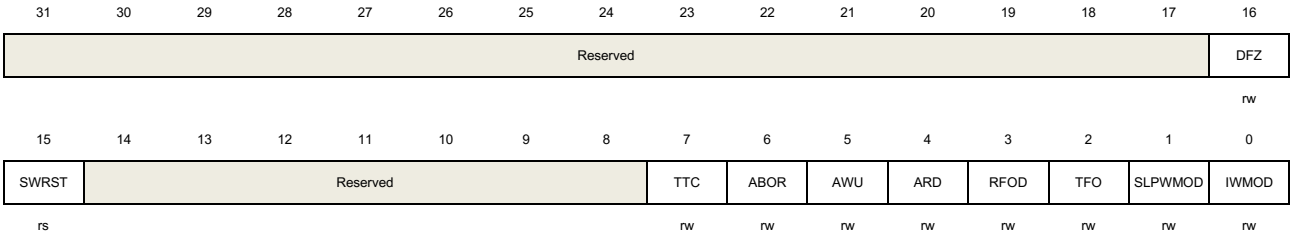
31.4. Register definition

CAN0 base address: 0x4000 6400
CAN1 base address: 0x4000 6800

31.4.1. Control register (CAN_CTL)

Address offset: 0x00
Reset value: 0x0001 0002

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



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	DFZ	Debug freeze If the CANx_HOLD in DBG_CTL register is set, this bit defines the CAN controller is in debug freezing mode or normal working mode. If the CANx_HOLD in DBG_CTL register is cleared, this bit takes no effect. 0: CAN reception and transmission work normal even during debug 1: CAN reception and transmission stop working during debug
15	SWRST	Software reset 0: No effect 1: Reset CAN to enter sleep working mode. This bit is automatically reset to 0.
14:8	Reserved	Must be kept at reset value.
7	TTC	Time-triggered communication 0: Disable time-triggered communication 1: Enable time-triggered communication
6	ABOR	Automatic Bus-Off recovery 0: The Bus-Off state is left manually by software 1: The Bus-Off state is left automatically by hardware
5	AWU	Automatic wakeup If this bit is set, the CAN leaves sleep working mode when CAN bus activity is detected, and SLPWMOD bit in CAN_CTL register will be cleared automatically. 0: The sleeping working mode is left manually by software

		1: The sleeping working mode is left automatically by hardware
4	ARD	Automatic retransmission disable 0: Enable automatic retransmission 1: Disable automatic retransmission
3	RFOD	Rx FIFO overwrite disable 0: Enable Rx FIFO overwrite when Rx FIFO is full and overwrite the FIFO with the incoming frame 1: Disable Rx FIFO overwrite when Rx FIFO is full and discard the incoming frame
2	TFO	Tx FIFO order 0: Order with the identifier of the frame (the smaller identifier has higher priority) 1: Order with first-in and first-out
1	SLPWMOD	Sleep working mode If this bit is set by software, the CAN enters sleep working mode after current transmission or reception is completed. This bit can be cleared by software or hardware. If AWU bit in CAN_CTL register is set, this bit is cleared by hardware when CAN bus activity is detected. 0: Disable sleep working mode 1: Enable sleep working mode
0	IWMOD	Initial working mode 0: Disable initial working mode 1: Enable initial working mode

31.4.2. Status register (CAN_STAT)

Address offset: 0x04

Reset value: 0x0000 0C02

This register 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				RXL	LASTRX	RS	TS	Reserved			SLPIF	WUIF	ERRIF	SLPWS	IWS
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	RXL	RX level
10	LASTRX	Last sample value of RX pin
9	RS	Receiving state

		0: CAN is not working in the receiving state 1: CAN is working in the receiving state
8	TS	Transmitting state 0: CAN is not working in the transmitting state 1: CAN is working in the transmitting state
7:5	Reserved	Must be kept at reset value.
4	SLPIF	Status change interrupt flag of entering sleep working mode This bit is set by hardware when entering sleep working mode, and cleared by hardware when the CAN is not in sleep working mode. This bit can also be cleared by software when writting 1 to this bit. 0: CAN is not in the sleep working mode 1: CAN is in the sleep working mode
3	WUIF	Status change interrupt flag of waking up from sleep working mode This bit is set when CAN bus activity event is detected in sleep working mode. This bit can be cleared by software when writting 1 to this bit. 0: Wakeup event is not coming 1: Wakeup event is coming
2	ERRIF	Error interrupt flag This bit is set by the following events. The BOERR bit in CAN_ERR register is set and BOIE bit in CAN_INTEN register is set. Or the PERR bit in CAN_ERR register is set and PERRIE bit in CAN_INTEN register is set. Or the WERR bit in CAN_ERR register is set and WERRIE bit in CAN_INTEN register is set. Or the ERRN bits in CAN_ERR register are set to 1 to 6 (not 0 and not 7) and ERRNIE in CAN_INTEN register is set. This bit is cleared by software when writting 1 to this bit. 0: No error interrupt event 1: Any error interrupt event has happened
1	SLPWS	Sleep working state This bit is set by hardware when the CAN enters sleep working mode after setting SLPWMOD bit in CAN_CTL register. If the CAN leaves normal working mode to sleep working mode, it must wait the current frame transmission or reception to be completed. This bit is cleared by hardware when the CAN leaves sleep working mode. Clear SLPWMOD bit in CAN_CTL register or automatically detect the CAN bus activity when AWU bit is set in CAN_CTL register. If leaving sleep working mode to normal working mode, this bit will be cleared after receiving 11 consecutive recessive bits from the CAN bus. 0: CAN is not in the state of sleep working mode 1: CAN is in the state of sleep working mode
0	IWS	Initial working state This bit is set by hardware when the CAN enters initial working mode after setting IWMOD bit in CAN_CTL register. If the CAN leaves normal working mode to initial working mode, it must wait the current frame transmission or reception to be

completed. This bit is cleared by hardware when the CAN leaves initial working mode after clearing IWMOD bit in CAN_CTL register. If leaving initial working mode to normal working mode, this bit will be cleared after receiving 11 consecutive recessive bits from the CAN bus.

0: CAN is not in the state of initial working mode

1: CAN is in the state of initial working mode

31.4.3. Transmit status register (CAN_TSTAT)

Address offset: 0x08

Reset value: 0x1C00 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMLS2	TMLS1	TMLS0	TME2	TME1	TME0	NUM[1:0]		MST2	Reserved			MTE2	MAL2	MTFNERR2	MTF2
r	r	r	r	r	r	r		rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MST1	Reserved			MTE1	MAL1	MTFNERR1	MTF1	MST0	Reserved			MTE0	MAL0	MTFNERR0	MTF0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

Bits	Fields	Descriptions
31	TMLS2	Transmit mailbox 2 last sending in Tx FIFO This bit is set by hardware when transmit mailbox 2 has the last sending order in the Tx FIFO with at least two frames pending.
30	TMLS1	Transmit mailbox 1 last sending in Tx FIFO This bit is set by hardware when transmit mailbox 1 has the last sending order in the Tx FIFO with at least two frames pending.
29	TMLS0	Transmit mailbox 0 last sending in Tx FIFO This bit is set by hardware when transmit mailbox 0 has the last sending order in the Tx FIFO with at least two frames pending.
28	TME2	Transmit mailbox 2 empty 0: Transmit mailbox 2 not empty 1: Transmit mailbox 2 empty
27	TME1	Transmit mailbox 1 empty 0: Transmit mailbox 1 not empty 1: Transmit mailbox 1 empty
26	TME0	Transmit mailbox 0 empty 0: Transmit mailbox 0 not empty 1: Transmit mailbox 0 empty
25:24	NUM[1:0]	These bits are the number of the Tx FIFO mailbox in which the frame will be transmitted if at least one mailbox is empty.

		These bits are the number of the Tx FIFO mailbox in which the frame will be transmitted at last if all mailboxes are full.
23	MST2	Mailbox 2 stop transmitting This bit is set by the software to stop mailbox 2 transmitting. This bit is reset by the hardware while the mailbox 2 is empty.
22:20	Reserved	Must be kept at reset value.
19	MTE2	Mailbox 2 transmit error This bit is set by hardware when the transmit error occurs. This bit is reset by writing 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
18	MAL2	Mailbox 2 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
17	MTFNERR2	Mailbox 2 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error. 0: Mailbox 2 transmit finished with error 1: Mailbox 2 transmit finished with no error
16	MTF2	Mailbox 2 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 1 to this bit or TEN bit in CAN_TMI2 is 1. 0: Mailbox 2 transmit is progressing 1: Mailbox 2 transmit finished
15	MST1	Mailbox 1 stop transmitting This bit is set by software to stop mailbox 1 transmitting. This bit is reset by hardware when the mailbox 1 is empty.
14:12	Reserved	Must be kept at reset value.
11	MTE1	Mailbox 1 transmit error This bit is set by hardware when the transmit error occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
10	MAL1	Mailbox 1 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
9	MTFNERR1	Mailbox 1 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset

by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error.
0: Mailbox 1 transmit finished with error
1: Mailbox 1 transmit finished with no error

8	MTF1	Mailbox 1 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 1 to this bit or TEN bit in CAN_TMI1 is 1. 0: Mailbox 1 transmit is progressing 1: Mailbox 1 transmit finished
7	MST0	Mailbox 0 stop transmitting This bit is set by the software to stop mailbox 0 transmitting. This bit is reset by the hardware when the mailbox 0 is empty.
6:4	Reserved	Must be kept at reset value.
3	MTE0	Mailbox 0 transmit error This bit is set by hardware when the transmit error occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
2	MAL0	Mailbox 0 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
1	MTFNERR0	Mailbox 0 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error. 0: Mailbox 0 transmit finished with error 1: Mailbox 0 transmit finished with no error
0	MTF0	Mailbox 0 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 1 to this bit or TEN bit in CAN_TMI0 is 1. 0: Mailbox 0 transmit is progressing 1: Mailbox 0 transmit finished

31.4.4. Receive message FIFO0 register (CAN_RFIFO0)

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										RFD0	RFO0	RFF0	Reserved	RFL0[1:0]	
										rs	rc_w1	rc_w1		r	

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	RFD0	Rx FIFO0 dequeue This bit is set by software to start dequeuing a frame from Rx FIFO0. This bit is reset by hardware when the dequeuing is done.
4	RFO0	Rx FIFO0 overfull This bit is set by hardware when Rx FIFO0 is overfull and reset by software when writting 1 to this bit. 0: The Rx FIFO0 is not overfull 1: The Rx FIFO0 is overfull
3	RFF0	Rx FIFO0 full This bit is set by hardware when Rx FIFO0 is full and reset by software when writting 1 to this bit. 0: The Rx FIFO0 is not full 1: The Rx FIFO0 is full
2	Reserved	Must be kept at reset value.
1:0	RFL0[1:0]	Rx FIFO0 length These bits are the length of the Rx FIFO0.

31.4.5. Receive message FIFO1 register (CAN_RFIFO1)

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										RFD1	RFO1	RFF1	Reserved	RFL1[1:0]	
										rs	rc_w1	rc_w1		r	

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	RFD1	Rx FIFO1 dequeue This bit is set by software to start dequeuing a frame from Rx FIFO1.

This bit is reset by hardware when the dequeuing is done.

4	RFO1	Rx FIFO1 overfull This bit is set by hardware when Rx FIFO1 is overfull and reset by writting 1 to this bit. 0: The Rx FIFO1 is not overfull 1: The Rx FIFO1 is overfull
3	RFF1	Rx FIFO1 full This bit is set by hardware when Rx FIFO1 is full and reset by writting 1 to this bit. 0: The Rx FIFO1 is not full 1: The Rx FIFO1 is full
2	Reserved	Must be kept at reset value.
1:0	RFL1[1:0]	Rx FIFO1 length These bits are the length of the Rx FIFO1.

31.4.6. Interrupt enable register (CAN_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														SLPWIE	WIE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			ERRNIE	BOIE	PERRIE	WERRIE	Reserved	RFOIE1	RFFIE1	RFNEIE1	RFOIE0	RFFIE0	RFNEIE0	TMEIE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	SLPWIE	Sleep working interrupt enable 0: Sleep working interrupt disabled 1: Sleep working interrupt enabled
16	WIE	Wakeup interrupt enable 0: Wakeup interrupt disabled 1: Wakeup interrupt enabled
15	ERRIE	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled
14:12	Reserved	Must be kept at reset value.
11	ERRNIE	Error number interrupt enable

		0: Error number interrupt disabled 1: Error number interrupt enabled
10	BOIE	Bus-Off interrupt enable 0: Bus-Off interrupt disabled 1: Bus-Off interrupt enabled
9	PERRIE	Passive error interrupt enable 0: Passive error interrupt disabled 1: Passive error interrupt enabled
8	WERRIE	Warning error interrupt enable 0: Warning error interrupt disabled 1: Warning error interrupt enabled
7	Reserved	Must be kept at reset value.
6	RFOIE1	Rx FIFO1 overfull interrupt enable 0: Rx FIFO1 overfull interrupt disabled 1: Rx FIFO1 overfull interrupt enabled
5	RFFIE1	Rx FIFO1 full interrupt enable 0: Rx FIFO1 full interrupt disabled 1: Rx FIFO1 full interrupt enabled
4	RFNEIE1	Rx FIFO1 not empty interrupt enable 0: Rx FIFO1 not empty interrupt disabled 1: Rx FIFO1 not empty interrupt enabled
3	RFOIE0	Rx FIFO0 overfull interrupt enable 0: Rx FIFO0 overfull interrupt disabled 1: Rx FIFO0 overfull interrupt enabled
2	RFFIE0	Rx FIFO0 full interrupt enable 0: Rx FIFO0 full interrupt disabled 1: Rx FIFO0 full interrupt enabled
1	RFNEIE0	Rx FIFO0 not empty interrupt enable 0: Rx FIFO0 not empty interrupt disabled 1: Rx FIFO0 not empty interrupt enabled
0	TMEIE	Transmit mailbox empty interrupt enable 0: Transmit mailbox empty interrupt disabled 1: Transmit mailbox empty interrupt enabled

31.4.7. Error register (CAN_ERR)

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
RECNT[7:0]								TECNT[7:0]							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									ERRN[2:0]			Reserved	BOERR	PERR	WERR
									rw				r	r	r

Bits	Fields	Descriptions
31:24	RECNT[7:0]	Receive error count defined by the CAN standard
23:16	TECNT[7:0]	Transmit error count defined by the CAN standard
15:7	Reserved	Must be kept at reset value.
6:4	ERRN[2:0]	Error number These bits indicate the error status of bit transformation. They are updated by hardware. When the bit transformation is successful, they are equal to 0. 000: No error 001: Stuff error 010: Form error 011: Acknowledgment error 100: Bit recessive error 101: Bit dominant error 110: CRC error 111: Set by software
3	Reserved	Must be kept at reset value.
2	BOERR	Bus-Off error Whenever the CAN enters Bus-Off state, the bit will be set by hardware.
1	PERR	Passive error Whenever the TECNT or RECNT is greater than 127, the bit will be set by hardware.
0	WERR	Warning error Whenever the TECNT or RECNT is greater than or equal to 96, the bit will be set by hardware.

31.4.8. Bit timing register (CAN_BT)

Address offset: 0x1C

Reset value: 0x0123 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCMOD	LCMOD	Reserved.	SJW[4:0]					Reserved	BS2[2:0]			BS1[3:0]			
rw	rw		rw						rw			rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	BS2[4:3]		BS1[6:4]			BAUDPSC[9:0]									
	rw		rw			rw									

Bits	Fields	Descriptions
31	SCMOD	Silent communication mode 0: Silent communication disabled 1: Silent communication enabled
30	LCMOD	Loopback communication mode 0: Loopback communication disabled 1: Loopback communication enabled
29	Reserved	Must be kept at reset value.
28:24	SJW[4:0]	Resynchronization jump width Resynchronization jump width time quantum = SJW[4:0]+1
23	Reserved	Must be kept at reset value.
22:20	BS2[2:0]	Bit segment 2 Bit segment 2 time quantum = BS2[4:0]+1
19:16	BS1[3:0]	Bit segment 1 Bit segment 1 time quantum = BS1[6:0]+1
15	Reserved	Must be kept at reset value.
14:13	BS2[4:3]	Bits 4:3 of BS2 See bits 22:20 of CAN_BT.(Configuration is valid when FDEN=1)
12:10	BS1[6:4]	Bits 6:4 of BS1 See bits 19:16 of CAN_BT. (Configuration is valid when FDEN=1)
9:0	BAUDPSC[9:0]	Baud rate prescaler The CAN baud rate prescaler

31.4.9. FD control register (CAN_FDCTL)

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									ESIMOD	TDCMOD	TDCEN	NISO	PRED	Reserved	FDEN
									rw	rw	rw	rw	rw		rw

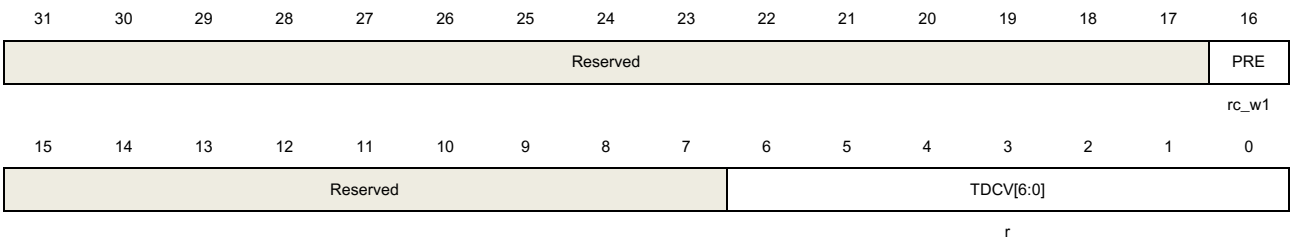
Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	ESIMOD	Error state indicator mode 0: Always displays the node error state. Transmit the dominant bit by error active nodes and transmit the recessive bit by error passive nodes. 1: When the node is not in the error passive state: Displays the message buffer error state by configuring ESI bit in CAN_TMPx registers. When the node is in the error passive state: Displays the node error state.
5	TDCMOD	Transmitter delay compensation mode 0: Measurement and offset 1: Only offset
4	TDCEN	Transmitter delay compensation enable 0: Transmitter delay compensation is disabled 1: Transmitter delay compensation is enabled
3	NISO	ISO/Bosch 0: ISO 1: Bosch
2	PRED	Protocol exception event detection disable 0: Protocol exception event detection enabled (to idle) 1: Protocol exception event detection disabled (regarded as a form error)
1	Reserved	Must be kept at reset value.
0	FDEN	FD operation enable 0: CAN FD function disabled 1: CAN FD function enabled

31.4.10. FD status register (CAN_FDSTAT)

Address offset: 0x24

Reset value: 0x0000 0000

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



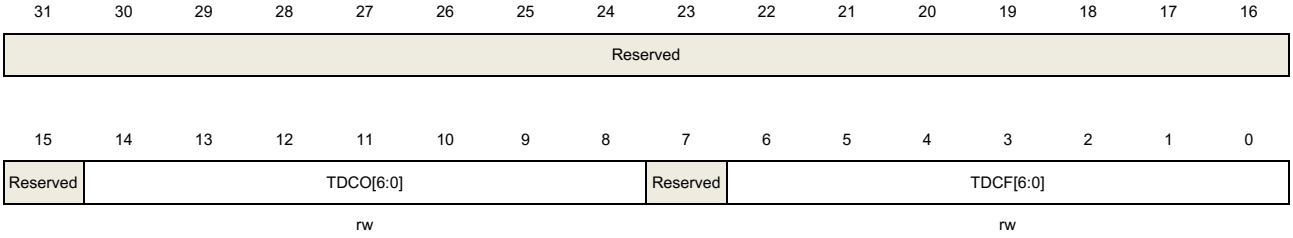
Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	PRE	Protocol exception event This bit is set by hardware when protocol exception event is detected, this bit is cleared by writting 1.
15:7	Reserved	Must be kept at reset value.
6:0	TDCV[6:0]	Transmitter delay compensation value These bits are set by hardware to display transmitter delay compensation value.

31.4.11. FD transmitter delay compensation register (CAN_FDTDC)

Address offset: 0x28

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.
14:8	TDCO[6:0]	Transmitter delay compensation offset 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.
7	Reserved	Must be kept at reset value.
6:0	TDCF[6:0]	Transmitter delay compensation filter These bits define the minimum value for the SSP position. Dominant edges on CANRX that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCV.

31.4.12. Date Bit timing register (CAN_DBT)

Address offset: 0x2C

Reset value: 0x0123 0000

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

Reserved	DSJW[2:0]	Reserved	DBS2[2:0]	DBS1[3:0]
	rw		rw	rw

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

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value.
25:24	DSJW[2:0]	Resynchronization jump width Resynchronization jump width time quantum = DSJW[2:0]+1
23	Reserved	Must be kept at reset value.
22:20	DBS2[2:0]	Bit segment 2 Bit segment 2 time quantum = DBS2[2:0]+1
19:16	DBS1[3:0]	Bit segment 1 Bit segment 1 time quantum = DBS1[3:0]+1
15:10	Reserved	Must be kept at reset value.
9:0	DBAUDPSC[9:0]	Baud rate prescaler The CAN baud rate prescaler

31.4.13. Transmit mailbox identifier register (CAN_TMIx) (x = 0...2)

Address offset: 0x180 + 0x10 * x
Reset value: 0xXXXX XXXX (bit0=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
SFID[10:0]/EFID[28:18]											EFID[17:13]				
rw											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFID[12:0]													FF	FT	TEN
rw													rw	rw	rw

Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:18]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier EFID[12:0]: Extended format frame identifier

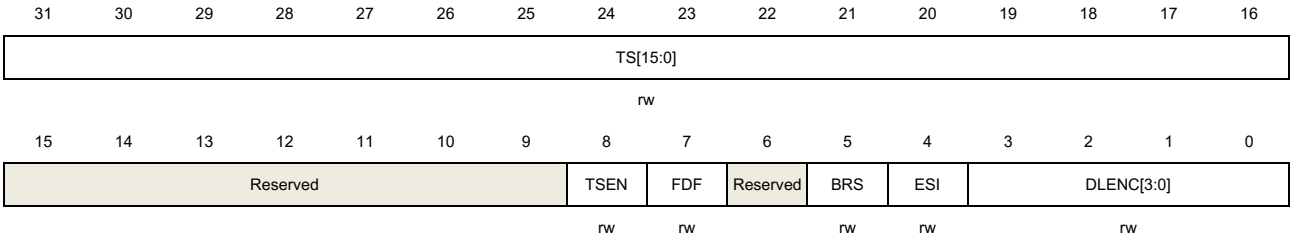
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	TEN	Transmit enable This bit is set by software when one frame will be transmitted and reset by hardware when the transmit mailbox is empty. 0: Transmit disabled 1: Transmit enabled

31.4.14. Transmit mailbox property register (CAN_TMPx) (x = 0…2)

Address offset: 0x184 + 0x10 * x

Reset value: 0xXXXX XXXX

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



Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:9	Reserved	Must be kept at reset value.
8	TSEN	Time stamp enable 0: Time stamp disabled 1: Time stamp enabled. The TS[15:0] will be transmitted in the DB6 and DB7 in DL. This bit is available when the TTC bit in CAN_CTL is set.
7	FDF	CAN FD frame flag 0: Classical frames 1: FD frames
6	Reserved	Must be kept at reset value.
5	BRS	Bit rate of data switch 0: Bit rate not switch 1: The bit rate shall be switched from the nominal bit rate of the arbitration phase to

		the preconfigured bit rate of data of the data phase
4	ESI	Error status indicator This bit is valid when ESIMOD bit is 1 in CAN_FDCTL register 0: Transmit the dominant bit in ESI phase 1: Transmit the recessive bit in ESI phase
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame.

31.4.15. Transmit mailbox data0 register (CAN_TMDATA0x) (x = 0…2)

Address offset: 0x188 + 0x10 * 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
DB3[7:0]								DB2[7:0]							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1[7:0]								DB0[7:0]							
rw								rw							

Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

31.4.16. Transmit mailbox data1 register (CAN_TMDATA1x) (x = 0...2)

Address offset: 0x18C + 0x10 * 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
DB7[7:0]								DB6[7:0]							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB5[7:0]								DB4[7:0]							
rw								rw							

Bits	Fields	Descriptions

31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

31.4.17. Rx FIFO mailbox identifier register (CAN_RFIFOMIx) (x = 0,1)

Address offset: 0x1B0 + 0x10 * x
Reset value: 0xXXXX XXXX

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



Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:18]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier EFID[12:0]: Extended format frame identifier
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	Reserved	Must be kept at reset value.

31.4.18. Rx FIFO mailbox property register (CAN_RFIFOMPx) (x = 0,1)

Address offset: 0x1B4 + 0x10 * x
Reset value: 0xXXXX XXXX

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

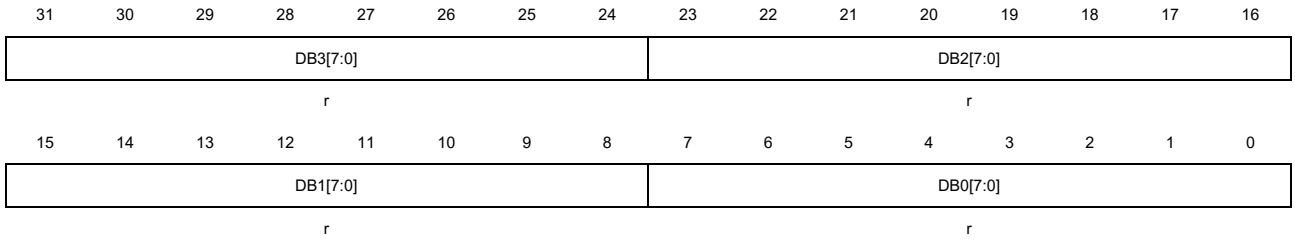


Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:8	FI[7:0]	Filtering index The index of the filter which the frame passes.
7	FDF	CAN FD frame flag 0: Classical frames 1: FD frames
6	Reserved	Must be kept at reset value.
5	BRS	Bit rate of data switch 0: Bit rate not switch 1: The bit rate shall be switched from the nominal bit rate of the arbitration phase to the preconfigured bit rate of data of the data phase
4	ESI	Error status indicator 0: Receive the dominant bit in ESI phase 1: Receive the recessive bit in ESI phase
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame.

31.4.19. Rx FIFO mailbox data0 register (CAN_RFIFOMDATA0x) (x = 0,1)

Address offset: 0x1B8 + 0x10 * x
Reset value: 0xXXXX XXXX

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



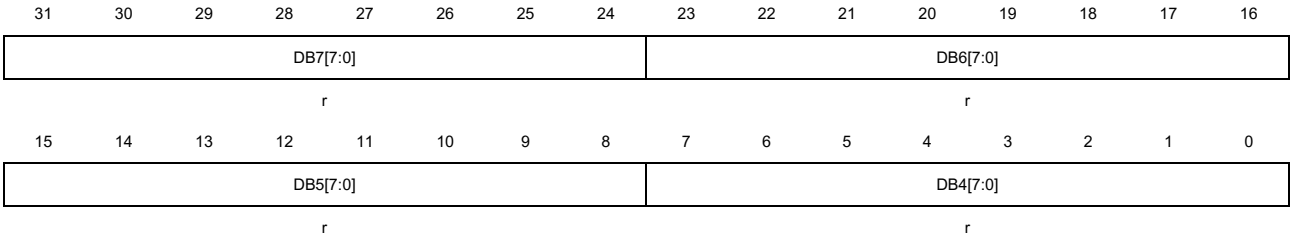
Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3

23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

31.4.20. Rx FIFO mailbox data1 register (CAN_RFIFOMDATA1x) (x=0,1)

Address offset: 0x1BC, 0x1CC
Reset value: 0xXXXX XXXX

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

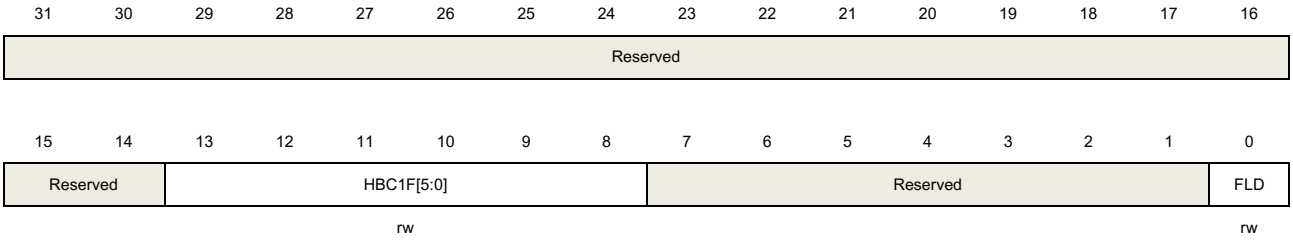


Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

31.4.21. Filter control register (CAN_FCTL)

Address: 0x40006600
Reset value: 0x2A1C 0E01

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



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:8	HBC1F[5:0]	Header bank of CAN1 filter These bits are set and cleared by software to define the first bank for CAN1 filter. Bank0 ~ Bank HBC1F-1 is used for CAN0. Bank HBC1F ~ Bank27 is used for CAN1.

When set to 0, no bank is used for CAN0. When set to 28, no bank is used for CAN1.

7:1	Reserved	Must be kept at reset value.
0	FLD	Filter lock disable 0: Filter lock enabled 1: Filter lock disabled

31.4.22. Filter mode configuration register (CAN_FMCFG)

Address: 0x40006604
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FMOD27	FMOD26	FMOD25	FMOD24	FMOD23	FMOD22	FMOD21	FMOD20	FMOD19	FMOD18	FMOD17	FMOD16
				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
FMOD15	FMOD14	FMOD13	FMOD12	FMOD11	FMOD10	FMOD9	FMOD8	FMOD7	FMOD6	FMOD5	FMOD4	FMOD3	FMOD2	FMOD1	FMOD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FMOD**x**	Filter mode 0: Filter **x** with mask mode 1: Filter **x** with list mode

31.4.23. Filter scale configuration register (CAN_FSCFG)

Address: 0x4000660C
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FS27	FS26	FS25	FS24	FS23	FS22	FS21	FS20	FS19	FS18	FS17	FS16
				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
FS15	FS14	FS13	FS12	FS11	FS10	FS9	FS8	FS7	FS6	FS5	FS4	FS3	FS2	FS1	FS0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.

27:0	FS**x**	Filter scale 0: Filter **x** with 16-bit scale 1: Filter **x** with 32-bit scale

31.4.24. Filter associated FIFO register (CAN_FAFIFO)

Address: 0x40006614
Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FAF27	FAF26	FAF25	FAF24	FAF23	FAF22	FAF21	FAF20	FAF19	FAF18	FAF17	FAF16
				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
FAF15	FAF14	FAF13	FAF12	FAF11	FAF10	FAF9	FAF8	FAF7	FAF6	FAF5	FAF4	FAF3	FAF2	FAF1	FAF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FAF**x**	Filter associated FIFO 0: Filter **x** associated with FIFO0 1: Filter **x** associated with FIFO1

31.4.25. Filter working register (CAN_FW)

Address: 0x4000661C
Reset value: 0x0000 0000

This register 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				FW27	FW26	FW25	FW24	FW23	FW22	FW21	FW20	FW19	FW18	FW17	FW16
				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
FW15	FW14	FW13	FW12	FW11	FW10	FW9	FW8	FW7	FW6	FW5	FW4	FW3	FW2	FW1	FW0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FW**x**	Filter working 0: Filter **x** working disable

1: Filter **x** working enable

31.4.26. Filter x data y register (CAN_FxDATAy) (x=0..27, y=0,1)

Address: 0x40006640 + 8 * x + 4 * y, (x = 0…27, y=0,1)

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
FD31	FD30	FD29	FD28	FD27	FD26	FD25	FD24	FD23	FD22	FD21	FD20	FD19	FD18	FD17	FD16
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
FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:0	FD**x**	Filter data Mask mode 0: Mask match disabled 1: Mask match enabled List mode 0: List identifier bit is 0 1: List identifier bit is 1

32. Ethernet (ENET)

32.1. Overview

This chapter describes the Ethernet peripheral module. There is a 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.

32.2. 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 to support standard frames with sizes up to 16 KB.
- Software configurable for inter-frame gap (40-96 bit times in steps of 8).
- 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 IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagram.
- 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 of 2K Byte for transmitting and receiving.
- Support special condition frame discards handling, e.g. late collision, excessive collisions, excessive deferral or underrun.
- Calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frame transmit under Store-and-Forward mode.

DMA Feature

- Two types of descriptor addressing: Ring and Chain.
- Each descriptor can transfer up to 8 KB of data.
- Programmable normal and abnormal interrupt for many status conditions.
- Round-robin or fixed-priority arbitration between reception and transmission controller.

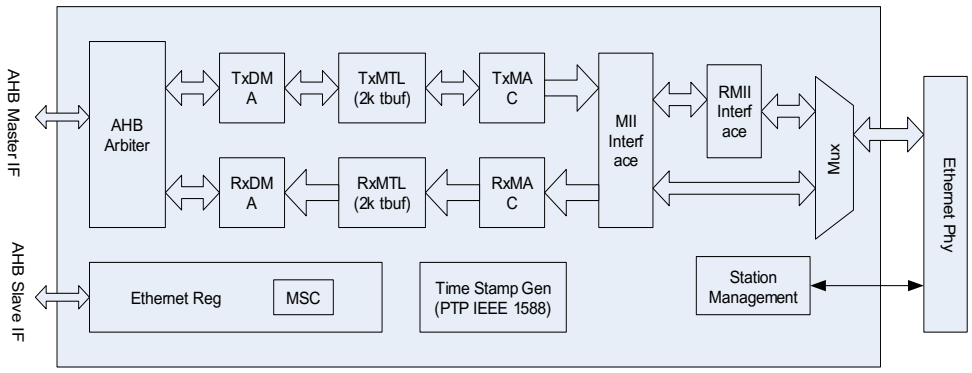
PTP Feature

- Support IEEE 1588 time synchronization function.
- Support two correction methods: Coarse or Fine.
- Pulse per second output.
- Preset target time reaching trigger and interrupt

32.2.1. Block diagram

The Ethernet module is composed of a MAC module, MII / RMII module and a DMA module by descriptor control.

Figure 32-1. ENET module block diagram



The MAC module is connected to the external PHY by MII or RMII through one selection bit (refer to SYSCFG_CFG1 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.

Note: The AHB clock frequency must be at least 25 MHz when the Ethernet is used.

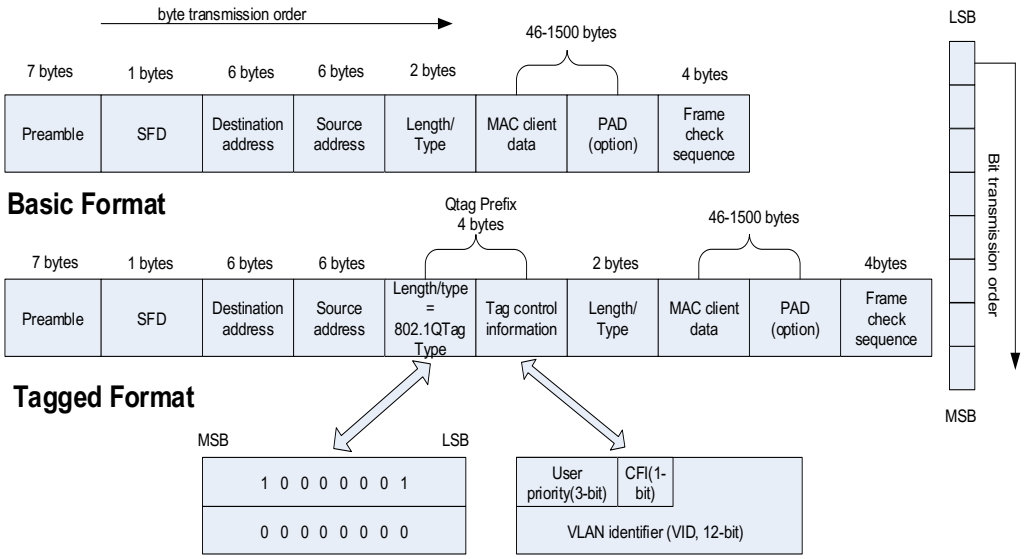
32.2.2. MAC 802.3 Ethernet packet description

Data communication of MAC can use two frame formats:

- Basic MAC frame format.
- Tagged MAC frame format (extension of the basic MAC frame format).

Figure 32-2. MAC / Tagged MAC frame format describes the structure of the frame (Basic and Tagged) that includes the following fields:

Figure 32-2. 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:

$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

32.2.3. Ethernet signal description

Table 32-1. Ethernet pin configuration shows the MAC module that pin is used default and remapping functions and specific configuration in MII / RMII mode.

Table 32-1. Ethernet pin configuration

MAC signals	PIN configure (AF11)	MII		RMII	
		PIN (1)	PIN (2)	PIN (1)	PIN (2)
ENET_MDC	Push-Pull Speed Level-3	PC1	-	PC1	-
ENET_MII_TXD2	Push-Pull Speed Level-3	PC2	-	-	-
ENET_MII_TX_CLK	Push-Pull Speed Level-3	PC3	-	-	-
ENET_MII_CRS	Push-Pull Speed Level-3	PA0	PH2	-	-
ENET_MII_RX_CLK ENET_RMII_REF_CLK	Push-Pull Speed Level-3	PA1	-	PA1	-
ENET_MDIO	Push-Pull Speed Level-3	PA2	-	PA2	-
ENET_MII_COL	Push-Pull Speed Level-3	PA3	PH3	-	-
ENET_MII_RX_DV ENET_RMII_CRS_DV	Push-Pull Speed Level-3	PA7	-	PA7	-
ENET_MII_RXD0 ENET_RMII_RXD0	Push-Pull Speed Level-3	PC4	-	PC4	-
ENET_MII_RXD1 ENET_RMII_RXD1	Push-Pull Speed Level-3	PC5	-	PC5	-
ENET_MII_RXD2	Push-Pull Speed Level-3	PB0	PH6	-	-
ENET_MII_RXD3	Push-Pull Speed Level-3	PB1	PH7	-	-
ENET_PPS_OUT	Push-Pull Speed Level-3	PB5	PG8	PB5	PG8
ENET_MII_TXD3	Push-Pull Speed Level-3	PB8	PE2	-	-
ENET_MII_RX_ER	Push-Pull Speed Level-3	PB10	PI10	-	-
ENET_MII_TX_EN ENET_RMII_TX_EN	Push-Pull Speed Level-3	PB11	PG11	PB11	PG11
ENET_MII_TXD0 ENET_RMII_TXD0	Push-Pull Speed Level-3	PB12	PG13	PB12	PG13

MAC signals	PIN configure (AF11)	MII		RMII	
		PIN (1)	PIN (2)	PIN (1)	PIN (2)
ENET_MII_TXD1 ENET_RMII_TXD1	Push-Pull Speed Level-3	PB13	PG14	PB13	PG14

Note: Application must be make sure only one PIN in PIN (1) and PIN (2) is configured to AF11 whichever interface mode (MII / RMII).

32.3. Function overview

32.3.1. 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

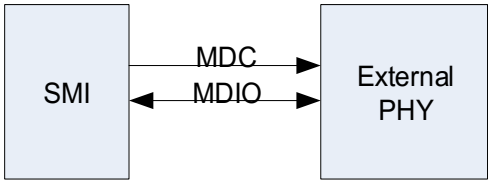
SMI is designed to access and configure PHY's configuration.

Station management interface (SMI) is performed through two wires to communicate with the external PHY: one clock line (MDC) and one data line (MDIO), it can access to the any PHY register. The interface supports accessing up to 32 PHYs, but only one register in one PHY can be addressed at the same time.

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 32-3. Station management interface signals



SMI 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.

SMI 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.

SMI clock selection

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 32-2. Clock range* lists the frequency factor corresponding AHB clock selection.

Table 32-2. Clock range

AHB clock	MDC clock	Selection
150~200MHz	AHB clock / 102	0x4
35~60MHz	AHB clock / 26	0x3
20~35MHz	AHB clock / 16	0x2
100~150 MHz	AHB clock / 62	0x1
60~100MHz	AHB clock / 42	0x0

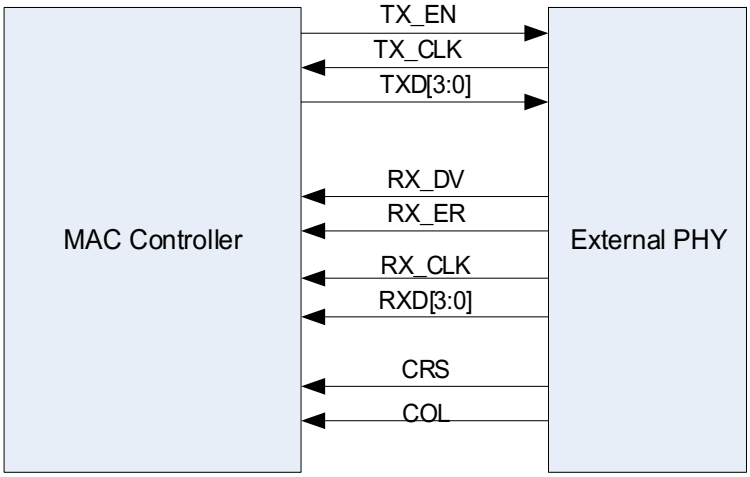
MII / RMII selection

The application can select the MII or RMII mode through the configuration bit 23 of the SYSCFG_CFG1 register ENET_PHY_SEL while the Ethernet controller is under reset state or before enabling the clocks. The MII mode is set by default.

MII: Media independent interface

The media-independent interface (MII) defines the interconnection between the MAC sub-layer and the PHY for data transfer at 10 Mbit / s or 100 Mbit / s.

Figure 32-4. 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. It must be asserted synchronously with the first bit of the preamble and must remain asserted while all bits to be transmitted are presented to the MII.

- **MII_TXD[3:0]**: Transmit data line, each 4 bit data transfer, data is valid when the MII_TX_EN signal is effective. MII_TXD[0] is the least significant bit and MII_TXD[3] is the most significant bit. While MII_TX_EN is de-asserted the transmit data must have no effect upon the PHY.

- **MII_CRS**: Carrier sense signal, only working in Half-duplex mode and controlled by the PHY. It is active when either transmit or receive medium is in non idle state. The PHY must ensure that the MII_CRS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the TX and RX clock.

- **MII_COL**: Collision detection signal, only working in Half-duplex mode, controlled by the PHY. It is active when a collision on the medium is detected and must it will remain active while the collision condition continues. This signal is not required to transition synchronously with respect to the TX and RX clock.

- **MII_RXD[3:0]**: Receive data line, each 4 bit data transfer; data are valid when the MII_RX_DV signal is effective. MII_RXD[0] is the least significant bit and MII_RXD[3] is the most significant bit. While MII_RX_DV is de-asserted and MII_RX_ER is asserted, a specific MII_RXD[3:0] value is used to indicate specific information (see ***Table 32-3. Rx interface***

signal encoding).

- **MII_RX_DV**: Receive data valid signal, controlled by the PHY. It is asserted when PHY is presenting data on the MII for reception. It must be asserted synchronously with the first 4-bit of the frame and must remain asserted while all bits to be transmitted are presented on the MII. It must be de-asserted prior to the first clock cycle that follows the final 4-bit. In order to receive the frame correctly, the effective signal starting no later than the SFD field.

- **MII_RX_ER**: Receive error signal. It must be asserted for one or more RX clock to indicate MAC detected an error in the receiving process. 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 32-3. Rx interface signal encoding*).

Table 32-3. Rx interface signal encoding

MII_RX_ER	MII_RX_DV	MII_RXD[3:0]	Description
0	0	0000 to 1111	Normal inter-frame
0	1	0000 to 1111	Normal reception frame data
1	0	0000	Normal inter-frame
1	0	0001 to 1101	Reserved
1	0	1110	False carrier indication
1	0	1111	Reserved
1	1	0000 to 1111	Data reception with errors

MII clock sources

To generate both TX_CLK and RX_CLK clock signals, the external PHY needs an external 25MHz 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_OUTx (x=0,1) pin. If the clock source is selected from CK_OUTx (x=0,1) pin, the MCU needs to configure the appropriate PLL to ensure the output frequency of CK_OUTx (x=0,1) pin is 25MHz.

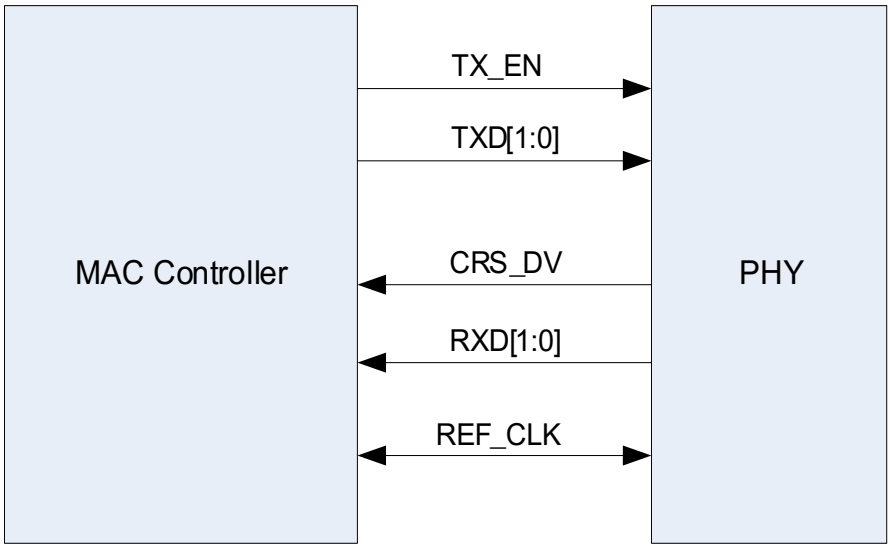
RMII: Reduced media independent interface

The reduced media-independent interface (RMII) specification reduces the pin count during Ethernet communication. According to the IEEE 802.3 standard, an MII contains 16 pins for data and control. The RMII specification is dedicated to reduce the pin count to 7.

The RMII block has the following 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 32-5. Reduced media-independent interface signals



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).
Transmission order for RMII use 4 cycles: 01 -> 11 -> 01 -> 10 (left to right order: high to low).

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_OUTx (x=0,1) pin. If the clock source is from CK_OUTx (x=0,1) pin, then the MCU needs to configure the appropriate PLL to ensure the output frequency of CK_OUTx (x=0,1) pin is 50MHz.

32.3.2. MAC function overview

MAC module can achieve the following functions:

Data package (transmission and reception)

- Frame detecting / decoding and frame boundary delimitation.
- Addressing (handling source address and destination address).
- Error conditions detect.

Medium access management in Half-duplex mode

- Medium allocation (prevent conflicts).

- Conflict resolution (dealing with conflicts).

The MAC module can work in two modes:

- 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.
- Full-duplex mode: simultaneous transmission and reception without any conflict mode, if all of the following conditions are satisfied:
  - PHY supports the feature of transmission and reception operations at the same time.
  - Only two devices connect to the LAN and the two devices are both configured for Full-duplex mode.

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, as soon as the number of bytes in the FIFO crosses or equals the configured threshold level or when the end-of-frame flag in descriptor is written, the data is ready to be popped out and forwarded to the MAC. The threshold level is configured using the TTHC bits of ENET_DMA_CTL.
- In Store-and-Forward mode, only after an integrated frame is stored in the FIFO, the frame is popped towards the MAC. But there is another condition for FIFO popping out data but the frame is not integrated. This is when the transmit FIFO size is smaller than the Ethernet frame to be transmitted, the frame is popped towards the MAC when the transmit FIFO becomes almost 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 status word

Transmit status word includes many transmit state flags for application and are updated after the complete the transmission of the frame. If timestamp function is enable, the timestamp value is also write back to transmit descriptor along with transmit status.

Transmit FIFO flush operation

Application can clear TxFIFO and reset the FIFO data pointer by setting FTF bit (bit 20) 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 (TDES0 bits 1 and 13). 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 flow control

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.

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.

Transmit checksum offload

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. This section describes the operation of the 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.

See 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 (bit 16 in TDES0). 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) . the frame ends before the IPv6 standard header or extension header (as given in the corresponding header length field in an extension header) has been completely received.

■ 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 TCP, UDP, or ICMP payload and inserts the result into its corresponding field in the header. It has two modes when working, as follows:

1. TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the input frame's checksum field. The checksum field is included in the checksum calculation, and then replaced by the final calculated checksum.
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 (bit 12 in TDES0). 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.

MAC receive filters

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

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.

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.

Hash or perfect address filter

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 by setting the HPFLT bit in the ENET_MAC_FRMF register and setting the corresponding HUF or HMF bit in the ENET_MAC_FRMF register.

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.

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 RDES0 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.

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 32-4. Destination address filtering table* and *Table 32-5. Source address*

filtering table summarize the destination address and source address filters working condition at different configuration.

Table 32-4. 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 32-5. 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 (DA + SA + LT + DATA + FCS) 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 (16K Bytes), 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.

Receive 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.

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 7 bit in RDES0) 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 (bit 0 in RDES0) 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.

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.

Receive status word

After receiving a complete frame, the MAC will analysis and record some state information about the frame and receiving process. These detail status information will write back to the receive descriptor and DMA status flag. Application can check these flags for upper protocol implementation.

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.

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.

32.3.3. 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 following situation, it can be called 'fine frame' and MSC transmit counters will automatically update:

- Frame underflow
- No carrier
- Lose of carrier
- Excessive deferral
- Late collision
- Excessive collision
- Jabber Timeout

When the receiving frame does not appear the following situation, it can be called 'fine frame' and MSC reception counters will automatically update:

- Alignment error
- CRC mismatch (calculated CRC value is different from FSC field value)
- Runt frame (frame length is shorter than 64 bytes)
- Length error (length field value is different from the actual received data bytes)
- Range error (length field value is larger than maximum size of defined in IEEE802.3, which is 1518 for untagged frame and 1522 for VLAN tagged frame)
- Error signal valid on pin MII_RX_ER

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.

32.3.4. 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.

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 32-6. 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.

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 19 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.

32.3.5. 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 correction method

The 64-bit PTP system time update by the PTP input reference clock. 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.

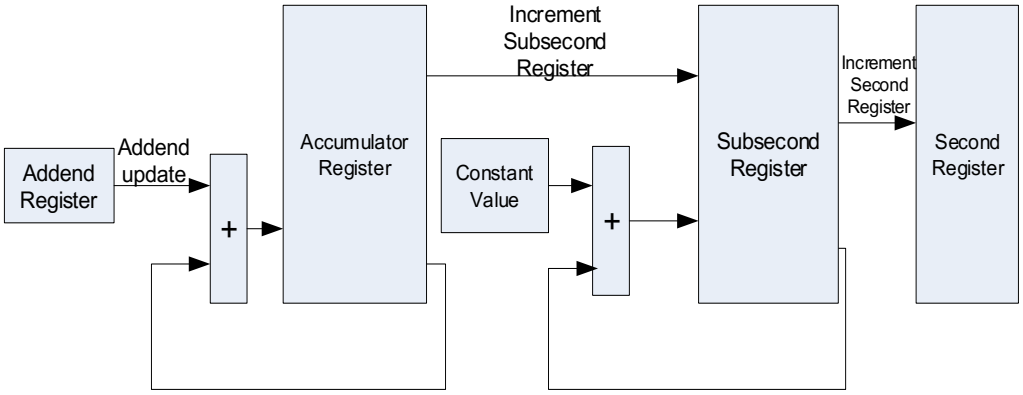
If the coarse correction method is selected, application can configure PTP timestamp update register (ENET_PTP_TSUH and ENET_PTP_TSUL) for system time initialization or correction. If TMSSTI bit is set, PTP timestamp update register is used for initialization and if TMSSTU bit is set, PTP timestamp update 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 32-7. System time update using the fine correction method shows the fine correction algorithm process:

Figure 32-7. 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 20ns, which means the frequency of update is 50MHz. If the reference clock of HCLK is 75MHz, the frequency ratio is calculated as 75 / 50, result is 1.5. Hence, the addend (TMSA bit in ENET_PTP_TSADDEND register) value to be set is 2^{32} / 1.5, which is equal to 0xAAAA AAAA. If the reference clock frequency drifts lower, for example, down to 65MHz, the frequency ratio changes to 65 / 50=1.3, the value to be set in the addend register is 2^{32} / 1.30 = 0xC4EC 4EC4. If the reference clock drift higher, for example, up to 85MHz, the value addend register must be 0xA000 0000. 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 20ns. The value of the register is to update values of timestamp low 32-bit register after accumulator register overflow. Because the timestamp low register (bit 0 to 30) represents the subsecond value of system time, the precision is 10^9ns / 2^{31}=0.46ns. So in order to make the system time accuracy to 20ns, sub second increment register value should be set to 20 / 0.46 = 0d43.

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 bit 9 in the ENET_MAC_INTMSK register to mask the timestamp trigger interrupt;
2. Setting bit 0 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 bit 5 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 bit 5 in the ENET_PTP_TSCTL register until it is cleared;
6. Set bit 1 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 bit 2 in the ENET_PTP_TSCTL register;
9. The timestamp counter starts counting as soon as the initialization process complete.

System time update steps under 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.

System time update steps under fine correction method

1. Calculate the value of the desired system clock rate corresponding to the addend register (***System time correction method*** has explained before);
2. Program the addend register, and set the bits 5 in ENET_PTP_TSCTL register;
3. Program the target high and low register and reset the bit 9 of the ENET_MAC_INTMSK register to allow time stamp interrupt;
4. Set bit 4 (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 bit 5 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 over Ethernet II 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" and "Receive DMA descriptor".

PTP trigger internal connection with TIMER1

MAC can provide trigger interrupt when the system time is no less than the target time. Using an interrupt imports a known latency and an uncertainty in the command execution time. In order to calculate the time of this known latency part, when the system time is greater than target time, the PTP module sets an output signal. Set bits[11:10] of TIMER1_IRMP register to 0x1 can make this signal internally connected to the ITI1 input of TIMER1. For this feature designed, no uncertainty is introduced because the clock of the TIMER1 and PTP reference clock (HCLK) are synchronous.

PTP pulse-per-second (PPS) output signal

Application configures ENET_PPS_OUT pin to AF11 to enable the PPS output function. This function can output a signal with the pulse width of 125ms by default (other width is detailed in register definition) 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 and connect them to one oscilloscope for clock measurement.

32.3.6. 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 following two kinds of data structures:

- Descriptor table (ring or chain type) and data buffer;
- Control and status register.

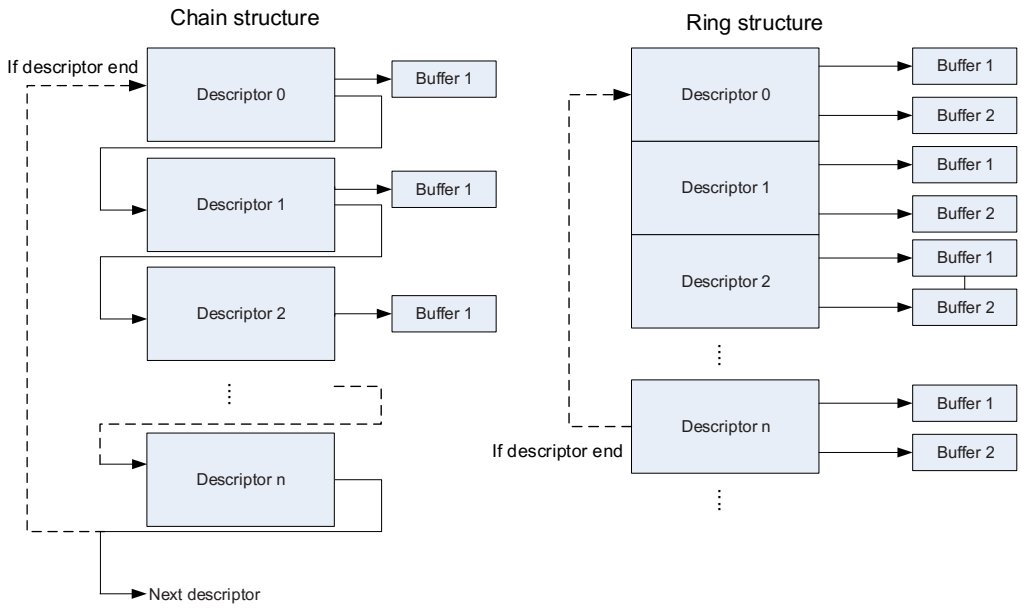
Applications need to provide the memory for storage of descriptor tables and data buffers. Descriptors that reside in the memory act as pointers to these buffers. Transmission has transmission descriptor and reception has reception descriptor. The base address of each table is stored in ENET_DMA_TDTADDR and ENET_DMA_RDTADDR register. Descriptors of transmission constituted by 4 descriptor word (TDES0-TDES3) when DFM=0 and 8 descriptor word (TDES0-TDES7) when DFM=1 (Enhanced descriptor mode). Likewise, reception descriptors constituted by 4 descriptor word (RDES0-RDES3) when DFM=0 and 8 descriptor word (RDES0-RDES7) 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 (RDES1[14] and TDES0[20]), at this time RDES2 and TDES2 are stored the data buffer address, RDES3 and TDES3 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 RDES1[14] and TDES0[20], at this time RDES2, TDES2 and RDES3, TDES3 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 bit 21 in TDES0 or bit 15 in RDES1 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 TDES3 or RDES3 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 32-8. 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 TDES1. 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 TDES0 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 TDES0 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 TDES0 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 RDES1. When the receive buffer address is not word align, the effective length of the buffer is less than the value configured by application in RDES1. 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 RDES0. 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.

Error response to DMA controller

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.

TxDMA configuration

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 (TDES0-3) with DAV bit of TDES0 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 TDES0[31] 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 TDES0[31] 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 TDES0[30] 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 TDES0[30] 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.

Transmit frame processing

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.

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

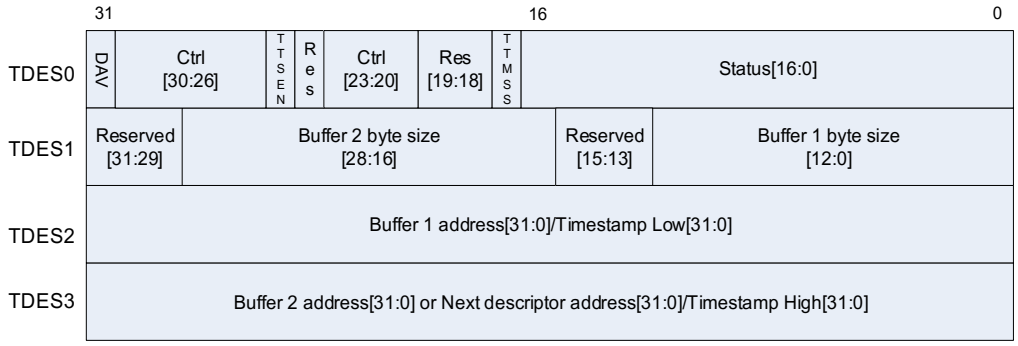
When TTSEN bit is set, the timestamp function is enabled. The TxDMA controller writes transmit timestamp status TTMSS and timestamp back to descriptor after the frame transmission complete. The word address in descriptor for writing timestamp is depends on DFM bit in ENET_DMA_BCTL register. If the descriptor format is normal mode (DFM=0), TDES2 and TDES3 are used for timestamp recording and the old values in TDES2 and TDES3 are overwritten. If the descriptor format is enhanced mode (DFM=1), TDES6 and TDES7 are used for timestamp recording and the value in TDES2 and TDES3 are kept.

TxDMA descriptors in normal mode

The normal mode descriptor structure consists of four 32-bit words: TDES0 ~ TDES3. The descriptions of TDES0, TDES1, TDES2 and TDES3 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 32-9. Transmit descriptor in normal mode



- TDES0: Transmit descriptor word 0

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 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. 0: The descriptor is available for CPU not for DMA

		1: The descriptor is available for DMA not for CPU
30	INTC	Interrupt on completion bit This is valid only when the last segment (TDES0[29]) is set. 0: TS bit in ENET_DMA_STAT is not set when frame transmission complete. 1: TS bit in ENET_DMA_STAT is set when frame transmission complete.
29	LSG	Last segment bit This bit indicates that the buffer contains the last segment of the frame. 0: The buffer of descriptor is not stored the last part of frame 1: The buffer of descriptor is stored the last part of frame
28	FSG	First segment bit This bit indicates that the buffer contains the first segment of a frame. 0: The buffer of descriptor is not stored the first block of frame 1: The buffer of descriptor is stored the first block of frame
27	DCRC	Disable CRC bit This is valid only when the first segment (TDES0[28]) is set. 0: The MAC automatic append a CRC to the end of the transmitted frame 1: The MAC does not append a CRC to the end of the transmitted frame
26	DPAD	Disable adding pad bit This is valid only when the first segment (TDES0[28]) is set. 0: The DMA automatically adds padding byte and CRC to a frame shorter than 64 bytes. Only the padding actually acts, the CRC is also appended. The DCRC bit is don't care. 1:The MAC does not automatically add padding to a frame
25	TTSEN	Transmit timestamp function enable bit. This field is only valid when the First segment control bit (TDES0[28]) is set. 0: Disable transmit timestamp function 1: When TMSEN is set (ENET_PTP_TSCTL bit 0), IEEE 1588 hardware time stamping is activated for the transmit frame.
24	Reserved	Must be kept at reset value.
23:22	CM[1:0]	Checksum mode bits 0x0: Disabled checksum insertion function 0x1: Only enable function for IP header checksum calculation and insertion 0x2: Enable IP header checksum and payload checksum calculation and insertion, pseudo-header checksum is not calculated in hardware. 0x3: Enable IP Header checksum and payload checksum calculation and insertion, pseudo-header checksum is calculated in hardware.
21	TERM	Transmit end for ring mode bit This bit is used only in ring mode and has higher priority than TCHM. 0: The current descriptor is not the last descriptor in the table

		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 This bit is used only in chain mode. When this bit, TCHM (TDES0[20]), is set, TB2S (TDES1[28:16]) is don't care. 0: The second address in the descriptor is the second buffer address 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 This bit is only valid when the descriptor's last segment (LSG) control bit (TDES0[29]) is set. 0: Timestamp was not captured 1:A timestamp was captured for the described transmit frame and push into TDES2 (or TDES6 if DFM=1) and TDES3 (or TDES7 if DFM=1)
16	IPHE	IP header error bit IP header error occurs when any case of below happen: IPv4 frames: 1) The header length field has a value less than 0x5. 2) The header length field value in transmitting IPv4 frame is mismatch with the number of header bytes. 3) The version field value does not match the length / type field value. IPv6 frames: 1) The main header length is not 40 bytes. 2) The version field value does not match the length / type field value. 0: The MAC transmitter did not detect error in the IP datagram header 1: The MAC transmitter detected an error in the IP datagram header
15	ES	Error summary bit Following bits are logical ORed to generate this bit: TDES0[16]: IP header error. TDES0[14]: Jabber timeout. TDES0[13]: Frame flush. TDES0[12]: IP payload error. TDES0[11]: Loss of carrier. TDES0[10]: No carrier. TDES0[9]: Late collision. TDES0[8]: Excessive collision. TDES0[2]: Excessive deferral. TDES0[1]: Underflow error.
14	JT	Jabber timeout bit Only set when the JBD bit is reset. 0: No jabber timeout occurred

		1: The MAC transmitter has experienced a jabber timeout
13	FRMF	Frame flushed bit This bit is set to flush the Tx frame by software.
12	IPPE	IP payload error bit 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. 0: No IP payload error occurred 1: MAC transmitter detected an error in the TCP, UDP, or ICMPIP datagram payload.
11	LCA	Loss of carrier bit When the interface signal 'CRS' lost one or more cycles and no collision happened during transmitting, the loss of carrier condition occurs. This is valid only in Half-duplex mode. 0: No loss of carrier occurred 1: A loss of carrier occurred during frame transmission
10	NCA	No carrier bit 0: PHY carrier sense signal is active 1: The carrier sense signal from the PHY was not asserted during transmission
9	LCO	Late collision bit If a collision occurs when 64 bytes (including preamble and SFD) has already transferred, this situation called late collision. 0: No late collision occurred 1: Late collision situation occurred **Note:** This bit is not valid if the UFE bit is set.
8	ECO	Excessive collision bit If the RTD=1 (retry function disable), this bit is set after the first collision. If the RTD=0 (retry function enable), this bit is set when failed 16 successive retry transmitting. When this bit is set, the transmission of current frame is aborted. 0: No excessive collision occurred 1: Excessive collision occurred
7	VFRM	VLAN frame bit 0: The transmitted frame was a normal frame 1: The transmitted frame was a VLAN-type frame
6:3	COCNT[3:0]	Collision count bits This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the ECO bit (TDES0[8]) is set.
2	EXD	Excessive deferral bit This is valid when the DFC bit in the MAC configuration register is set.

0: No excessive deferral occurred
1: The transmission has ended because of excessive deferral time is over 3036 bytes

1	UFE	Underflow error bit This bit indicates 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. 0: No underflow error occurred 1: Underflow error occurred and the MAC aborted the frame transmitting
0	DB	Deferred bit This bit indicates whether the transmitting frame is deferred because of interface signal CRS is active before MAC transmit frame. Valid only in Half-duplex mode. 0: No transmission deferred 1:The MAC is deferred before transmission

- TDES1: Transmit descriptor word 1

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 These bits indicate byte size of the second data buffer.
15:13	Reserved	Must be kept at reset value.
12:0	TB1S[12:0]	Transmit buffer 1 size bits These bits indicate the byte size of the first data buffer.

- TDES2: Transmit descriptor word 2

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 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.

- TDES3: 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															

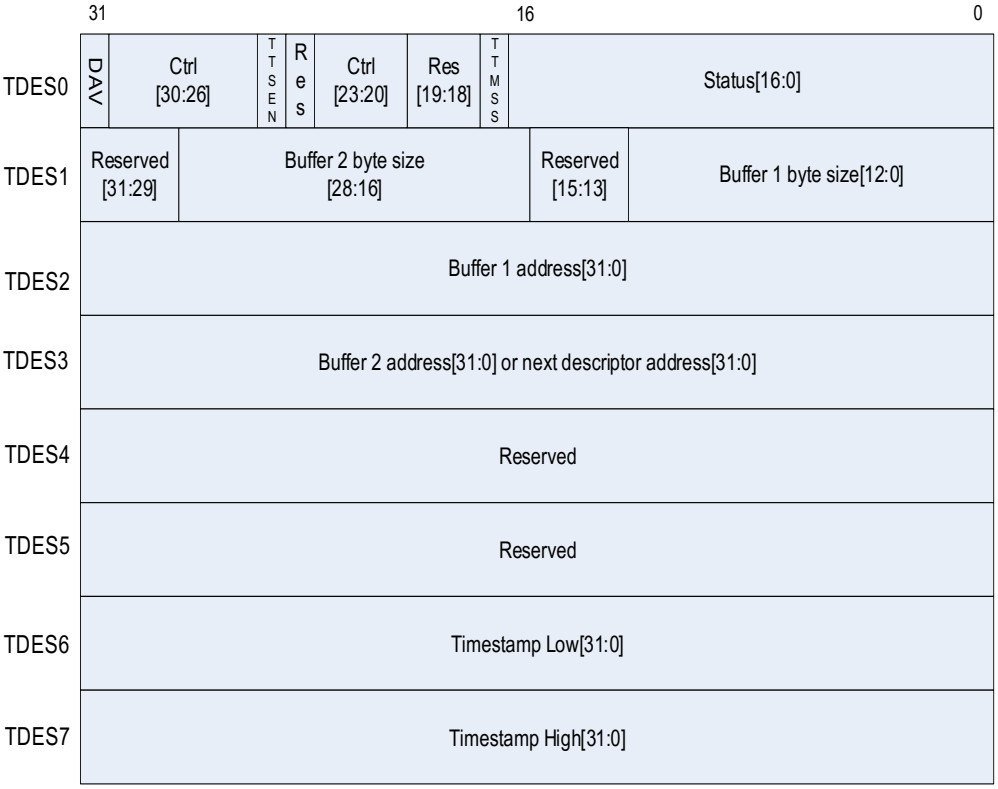
Bits	Fields	Descriptions
31:0	TB2AP / TTSH[31:0]	Transmit buffer 2 address pointer (or next descriptor address) / Transmit frame timestamp high 32-bit value bits. Before transmitting frame, application must configure these bits for transmit buffer 2 address (TB2AP) or the next descriptor address which 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

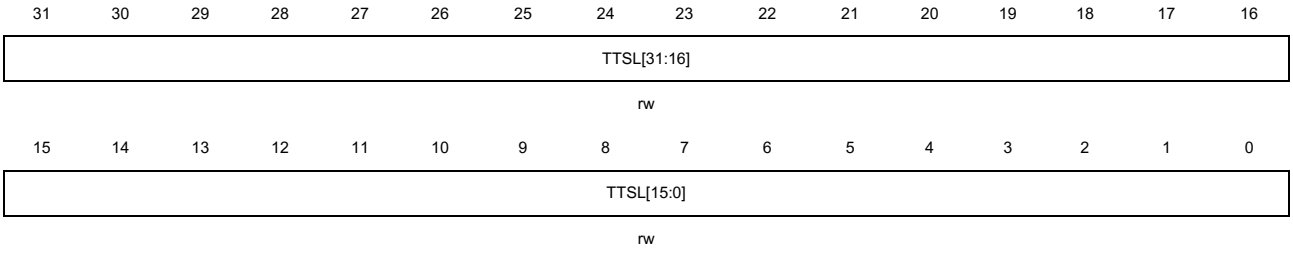
The enhanced mode descriptor structure consists of eight 32-bit words: TDES0 ~ TDES7. The descriptions of TDES0, TDES1, TDES2 and TDES3 are the same with normal mode descriptor; TDES4, TDES5, TDES6 and TDES7 are given below:

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 32-10. Transmit descriptor in enhanced mode

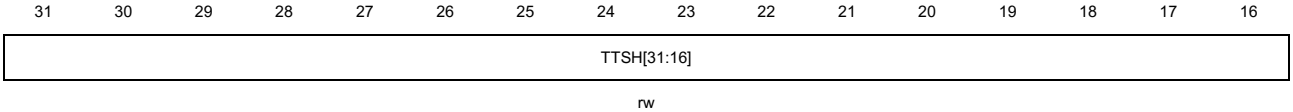


- TDES4 Transmit descriptor word 4
  All bits reserved.
- TDES5 Transmit descriptor word 5
  All bits reserved.
- TDES6 Transmit descriptor word 6



Bits	Fields	Descriptions
31:0	TTSL[31:0]	Transmit frame timestamp low 32-bit value bits When TTSEN =1 and LSG=1, there bits are updated by TxDMA for recording timestamp low 32-bit value of the current transmitting frame.

- TDES7 Transmit descriptor word 7



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 When TTSEN =1 and LSG=1, these bits are updated by TxDMA for recording timestamp high 32-bit value of the current transmitting frame.

RxDMA configuration

The receiving process of the RxDMA controller is described detailed as below:

1. Applications initialize the receive descriptors with the DAV bit (RXDES0[31]) 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 RDES0 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 RDES2 and RDES3 if DFM=0 or RDES6 and RDES7 if DFM=1. At the same time (writing timestamp value) the RxDMA controller also writes the received frame's status word to the RDES0 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.

Process of receiving frame

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 RDES0 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 RDES0 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 RDES0 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.

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 RDES0 is set, the RxDMA controller exits suspend state and returns to running state for frame reception. But if the DAV bit in RDES0 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.

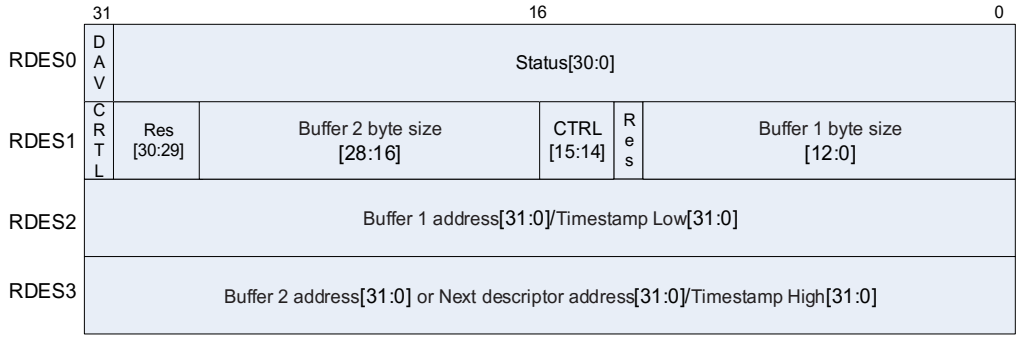
Receive DMA descriptor with IEEE 1588 timestamp format

If the IEEE 1588 function enabled, the MAC writes the timestamp value to RDES2 and RDES3 (DFM=0) or RDES6 and RDES7 (DFM=1) after a frame with timestamp reception complete and before the RxDMA controller clears the DAV bit.

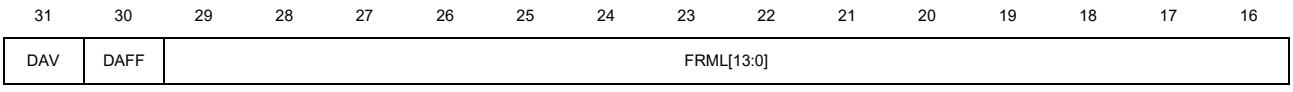
RxDMA descriptors in normal mode

In normal descriptor mode, the descriptor structure consists of four 32-bit words: RDES0 ~ RDES3. The detailed description of RDES0, RDES1, RDES2 and RDES3 are given below.

Figure 32-11. Receive descriptor in normal mode



- RDES0: Receive descriptor word 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 This bit indicates 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. 0: The descriptor is owned by the CPU 1: The descriptor is owned by the DMA
30	DAFF	Destination address filter fail bit 0: A frame passed the destination address filter 1: A frame failed the destination address filter
29:16	FRML[13:0]	Frame length bits These bits indicate 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). This field is valid only when LDES=1 (RDES0[8]) and DERR=0 (RDES0[14]). If LDES=0 and ERRS=0, these bits indicate the accumulated number of bytes that have been transferred for the current frame.
15	ERRS	Error summary bit This field is valid only when the LDES (RDES0[8]) is set. This bit is logical ORed by the following bits when DFM is equal to 0: RDES0[14]: Descriptor error. RDES0[11]: Overflow error RDES0[6]: Late collision RDES0[4]: Watchdog timeout RDES0[3]: Receive error RDES0[1]: CRC error REDS0[7] = 0, REDS0[5] = 1 and REDS0[0] = 1: payload checksum error. REDS0[7] = 1, REDS0[5] = 1 and REDS0[0] = 0: header checksum error. REDS0[7] = 1, REDS0[5] = 1 and REDS0[0] = 1: both header and payload checksum errors. This bit is logical ORed by the following bits when DFM is equal to 1: REDS4[4]: IP frame payload error REDS4[3]: IP frame header error RDES0[14]: Descriptor error RDES0[11]: Overflow error RDES0[6]: Late collision

		RDES0[4]: Watchdog timeout RDES0[3]: Receive error RDES0[1]: CRC error
14	DERR	Descriptor error bit This field is valid only when the LDES (RDES0[8]) is set. When the current buffer cannot hold current received frame and the next descriptor's DAV bit is reset, the descriptor error occurs. 0: No descriptor error occurred 1: Descriptor error occurred
13	SAFF	SA filtering fail bit 0: No source address filter fail occurred 1: A received frame failed the SA filter
12	LERR	Length error bit This bit is valid only when the FRMT (RDES0[5]) bit is reset. This bit indicates the mismatch between the length field in received and the actual frame length. 0: No length error occurred 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. 0: No overflow error occurred 1: RxFIFO overflowed and frame data is not valid
10	VTAG	VLAN tag bit 0: Received frame is not a tag frame 1: Received frame is a tag frame
9	FDES	First descriptor bit This bit indicates that current descriptor contains the SOF of the received frame. 0: The current descriptor does not store the SOF of the received frame 1: The current descriptor buffer saves the SOF of the received frame
8	LDES	Last descriptor bit This bit indicates that current descriptor contains the EOF of the received frame. 0: The current descriptor buffer does not store EOF of the received frame 1: The current descriptor buffer saves the EOF of the received frame
7	IPHERR / TSV	IP frame header error bit / Timestamp 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 timestamp value is taken and write to the RDES6 and RDES7. This bit is valid only when LDES is set.
6	LCO	Late collision bit

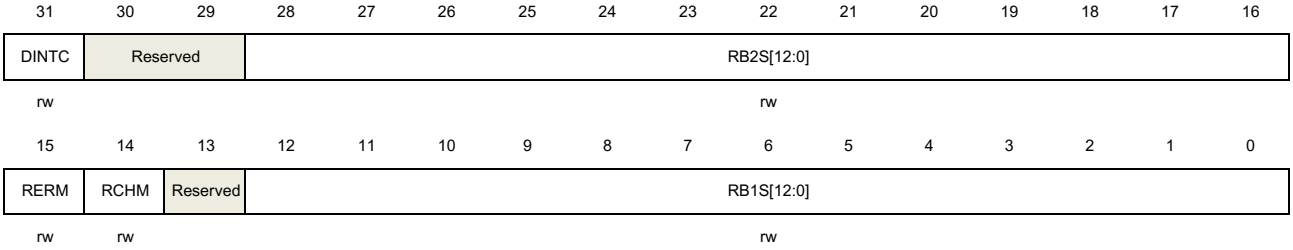
		This bit indicates a collision occurs after 64 bytes have been received. This bit only valid in Half-duplex mode. 0: No late collision occurred 1: Late collision has occurred
5	FRMT	Frame type bit When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table. When DFM=1, this bit indicates the received frame is an Ethernet type frame or a tagged frame. If the received frame is runt frame, this bit is not valid for application. 0: The received frame is an IEEE802.3 frame without tagged. 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 When WDD=0, this bit indicates a frame with more than 2048 bytes was detected. When WDD=1, this bit indicates a frame with more than 16384 bytes was detected. 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 indicates 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 is valid only in MII interface mode and indicates there is an incomplete byte (odd cycles during reception) received. 0: No dribble bit error occurred 1: Dribble bit error occurred
1	CERR	CRC error bit This bit is valid only when the LDES (RDES0[8]) is set and indicates FCS field in received frame is mismatch with the calculation result of the hardware. 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 RDES4 is valid for application. This bit only valid when LDES is set. 0: RDES4 is not valid for application 1:RDES4 is valid for application

Table 32-6. Error status decoding in RDES0, only used for normal descriptor (DFM=0) shows the combination meaning for bit 7, 5, and 0 in RDES0:

Table 32-6. Error status decoding in RDES0, 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. This error may case by following condition: 1) Calculated checksum value mismatch the checksum field 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. Offload check engine is bypassed.
1	1	0	IPv4 or IPv6 frame, but a header checksum error detected This error may case by following condition: 1) Type value inconsistent with version value 2) Calculated header checksum mismatch the header checksum field 3) Expected IP header bytes is not received enough
1	1	1	IPv4 or IPv6 frame, both header and payload checksum detected errors

- RDES1: Receive descriptor word 1



Bits	Fields	Descriptions
31	DINTC	Disable interrupt on completion bit 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. 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 The second buffer size in bytes. The buffer size must be a multiple of 4. This field is ignored if RCHM (RDES1[14]) is set.
15	RERM	Receive end of ring mode bit This bit indicates the final descriptor in table is arrived and the next descriptor address is automatically set to the configured start descriptor address. 0: Current descriptor is not the last descriptor in table 1: Current descriptor is the last descriptor in table
14	RCHM	Receive chained mode for second address bit 0: The second address points to the second buffer address. 1: The second address points to the next descriptor address. RB2S (RDES1[28:16]) 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 The first buffer size in bytes. The buffer size must be a multiple of 4.

- RDES2: Receive descriptor word 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 These bits are designed for two different functions: buffer address pointer (RB1AP) or timestamp low 32-bit value (RTSL). 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. 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 snapshoot condition. If the received frame does not meet the snapshoot condition, these bits will keep RB1AP value.

- RDES3: Receive descriptor word 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RB2AP / RTSH[31:16]															

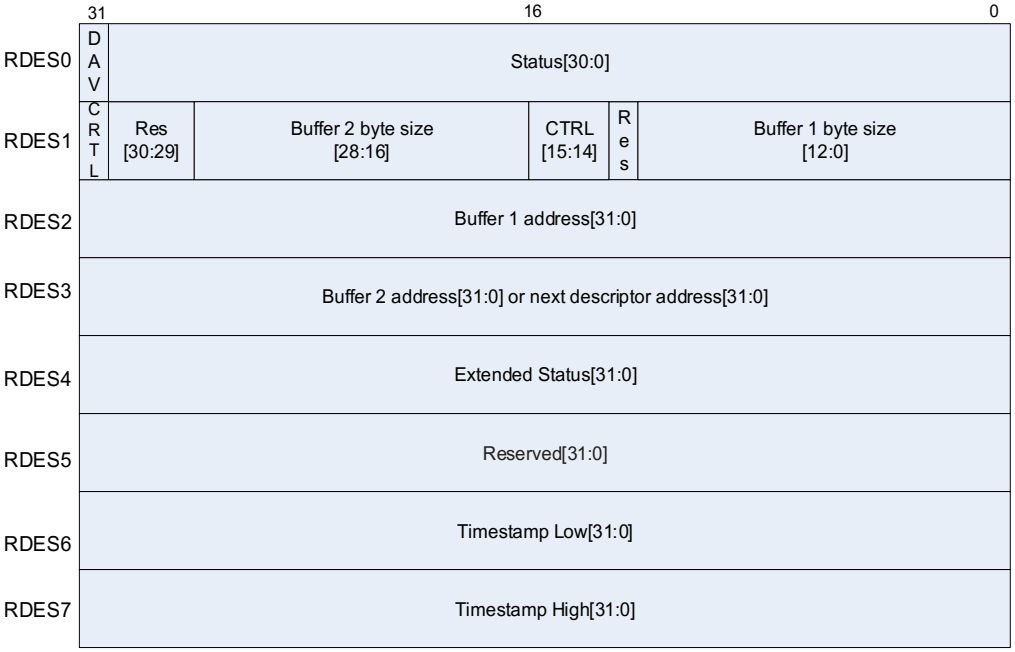
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB2AP / RTSH[15:0]															
rw															

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 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. 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 snapshoot condition, these bits will keep RB2AP value.

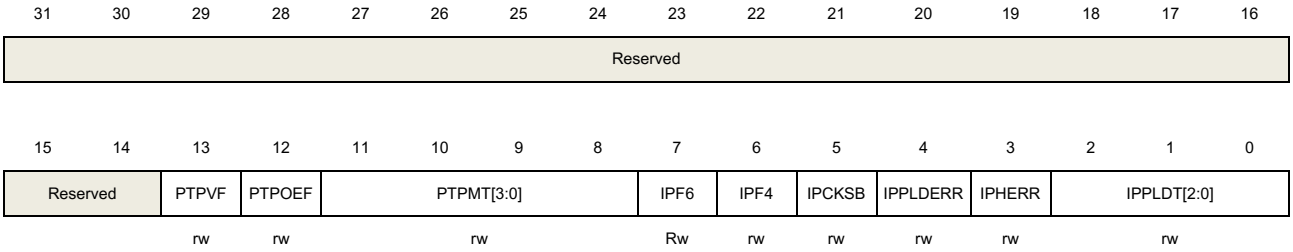
RxDMA descriptors in enhanced mode

In enhanced descriptor mode, the descriptor structure consists of eight 32-bit words: RDES0 ~ RDES7. The description of RDES0, RDES1, RDES2 and RDES3 are the same with descriptors in normal mode. The description of RDES4, RDES5, RDES6, and RDES7 are given below.

Figure 32-12. Receive descriptor in enhanced mode



- RDES4: Receive descriptor word 4



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	PTPVF	PTP version format bit 0: Version 1 format 1: Version 2 format
12	PTPOEF	PTP on Ethernet frame bit 0: Received PTP frame is a IP-UDP frame if PTPMT is not zero 1: Received PTP frame is a IEEE802.3 Ethernet frame
11:8	PTPMT[3:0]	PTP message type bits PTP message type is decoded to following number: 0x0: Not PTP frame received 0x1: SYNC 0x2: FOLLOW_UP 0x3: DELAY_REQ 0x4: DELAY_RESP 0x5: For peer-to-peer transparent clock: PDELAY_REQ For ordinary or boundary clock: ANNOUNCE

		0x6: For peer-to-peer transparent clock: PDELAY_RESP For ordinary or boundary clock: MANAGEMENT 0x7: For peer-to-peer transparent clock: PDELAY_RESP_FOLLOW_UP For ordinary or boundary clock: SIGNALING
7	IPF6	IP frame in version 6 bit 0: Received frame is not a IPv6 frame 1: Received frame is a IPv6 frame
6	IPF4	IP frame in version 4 bit 0: Received frame is not a IPv4 frame 1: Received frame is a IPv4 frame
5	IPCKSB	IP frame checksum bypassed bit This bit is only valid when received frame is a IPv4 or IPv6 frame. 0: Received frame checksum checking function is not bypassed 1: Received frame checksum checking function is bypassed
4	IPPLDERR	IP frame payload error bit 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. 0: Payload error not occurred in received frame 1: Payload error occurred in received frame
3	IPHERR	IP frame header error bit 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). 0: IP header error not occurred 1: IP header error occurred
2:0	IPPLDT[2:0]	IP frame payload type bits These bits are valid only when IPFCO=1, IPHERR=0 and LDES=1. 0x0: Unsupported payload type or IP payload bypassed 0x1: payload type is UDP 0x2: payload type is TCP 0x3: payload type is ICMP 0x4~0x7: Reserved

- RDES5: Receive descriptor word 5

  All bits reserved

- RDES6: Receive descriptor word 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 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.

- RDES7: Receive descriptor word 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 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.

32.3.7. 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 module to define which interface mode is selected (MII or RMII). Configure GPIO module to make selected PADs to alternate function 11(AF11).

- **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 TDES;
  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 RDES0 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.

32.3.8. 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.

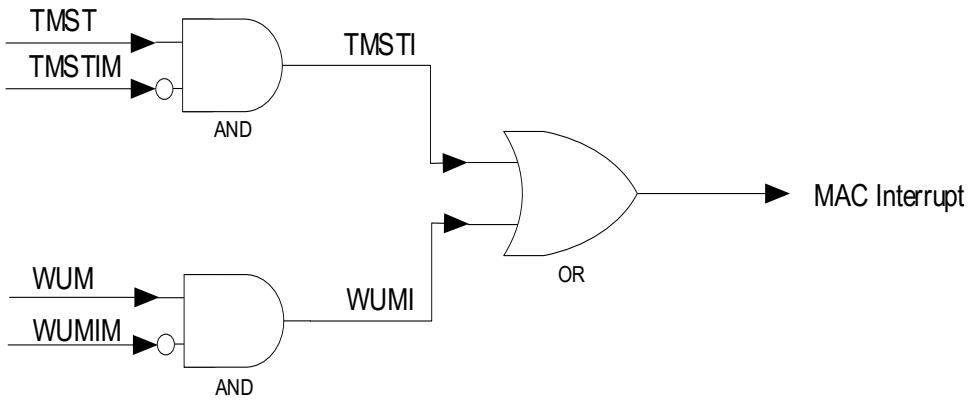
The WUM block event is connected to the second interrupt vector. The event can be remote wakeup frame received event or / and Magic Packet wakeup frame received event. This interrupt is inner mapped on the EXTI line 19. So, if the EXTI line 19 is enabled and configured to trigger by rising edge, the 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 32-13. MAC interrupt scheme



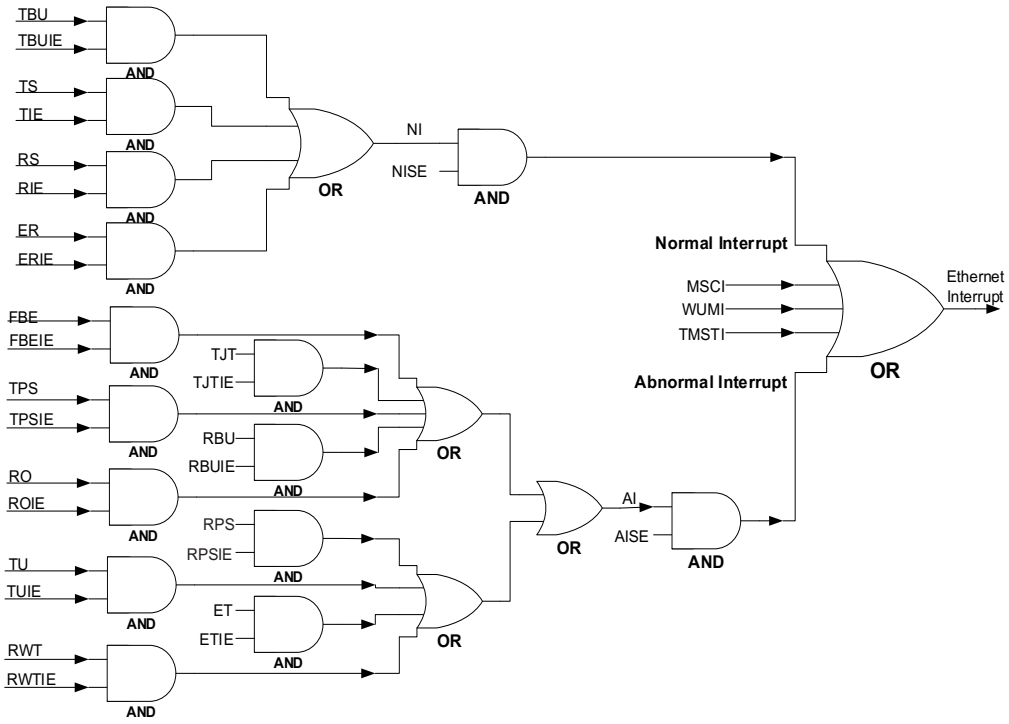
DMA controller interrupts

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 32-14. Ethernet interrupt scheme shows the Ethernet module interrupt connection:

Figure 32-14. 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

32.4. Register definition

ENET base addresss: 0x4002 8000
Byte (8-bit) access, half word (16-bit) access and word (32-bit) access are all supported for application.

32.4.1. 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 0: FCS field (last 4 bytes) of frame will not be dropped before forwarding 1: FCS field (last 4 bytes) of frame will be dropped before forwarding **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 This bit indicates the maximum bytes for receiving, data beyond this will be cut off. 0: The MAC allows no more than 2048 bytes of the frame being received 1: The MAC disables the watchdog timer on the receiver, and can receive frames of up to 16384 bytes
22	JBD	Jabber disable bit This bit indicates the maximum bytes for transmitting data, data beyond this will be cut off. 0: The maximum transmission byte is 2048 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 selection bits

		These bits can select the minimum inter frame gap time between two neighboring frames during transmission. The transmission clock is 25M at 100Mbps and 2.5M at 10Mbps 0x0: 96 transmission clock cycles 0x1: 88 transmission clock cycles 0x2: 80 transmission clock cycles 0x3: 72 transmission clock cycles 0x4: 64 transmission clock cycles 0x5: 56 transmission clock cycles (For Half-duplex, must be reserved) 0x6: 48 transmission clock cycles (For Half-duplex, must be reserved) 0x7: 40 transmission clock cycles (For Half-duplex, must be reserved)
16	CSD	Carrier sense disable bit 0: The MAC transmitter generates carrier sense error and aborts the transmission 1: The MAC transmitter ignores the MII CRS signal during frame transmission in Half-duplex mode. 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 Indicates the speed in Fast Ethernet mode: 0: 10 Mbit / s 1: 100 Mbit / s
13	ROD	Receive own disable bit This bit is not applicable if the MAC is operating in Full-duplex mode. 0: The MAC receives all packets that are given by the PHY while transmitting 1: The MAC disables the reception of frames in Half-duplex mode
12	LBM	Loopback mode bit 0: The MAC operates in normal mode 1: The MAC operates in loopback mode at the MII.
11	DPM	Duplex mode bit 0: Half-duplex mode enable 1: Full-duplex mode enable
10	IPFCO	IP frame checksum offload bit 0: The checksum offload function in the receiver is disabled 1: IP frame checksum offload function enabled for received IP frame
9	RTD	Retry disable bit This bit is applicable only in the Half-duplex mode. 0: The MAC attempts retries up to 16 times based on the settings of BOL 1: The MAC attempts only 1 transmission.
8	Reserved	Must be kept at reset value.
7	APCD	Automatic pad / CRC drop bit

		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 1: The MAC strips the Pad / FCS field on received frames
6:5	BOL[1:0]	Back-off limit bits 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: $0<=dt<2^{k}$. 0x0: k = min (n, 10) 0x1: k = min (n, 8) 0x2: k = min (n, 4) 0x3: k = min (n, 1), n = number of times for retransmission attempt **Note:** This bit is valid only in Half-duplex mode.
4	DFC	Deferral check bit 0: The deferral check function is disabled. MAC defers sending until the CRS goes inactive. 1: The deferral check function is enabled in the 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. **Note:** This bit is valid only in Half-duplex mode.
3	TEN	Transmitter enable bit 0: The MAC transmit function is disabled after finish the transmission of the current frame, and no frames to be transmitted anymore. 1: The transmit function of the MAC is enabled for transmission
2	REN	Receiver enable bit 0: The MAC reception function is disabled after finish the reception of the current frame, and no frames will be received anymore. 1: The MAC reception function is enabled for receiving frames
1:0	Reserved	Must be kept at reset value.

32.4.2. 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 receive bit This bit controls the receive filter function. 0: Only the frame passed the filter can be forwarded to application 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 0: If the HUF or HMF bit is set, only frames that match the hash filter are passed. 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 Enable source address filtering function besides destination address filtering. 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. 0: Source address function in filter disable 1: Source address function in filter enable
8	SAIFLT	Source address inverse filtering bit This bit makes the result of SA matching inverse. 0: Not inverse for source address filtering 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 These bits set the forwarding conditions for all control frames (including unicast and multicast pause frame). For pause control frame, the processing (not forwarding) depends only on RFCEN in ENET_MAC_FCTL[2]. 0x0: MAC prevents all control frames from reaching the application 0x1: MAC only forwards all other control frames except pause control frame 0x2: MAC forwards all control frames to application even if they fail the address filter 0x3: MAC forwards control frames that only pass the address filter
5	BFRMD	Broadcast frames disable bit 0: The address filters pass all received broadcast frames 1:The address filters filter all incoming broadcast frames

4	MFD	Multicast filter disable bit 0: Filtering of multicast frame depends on the HMF bit 1: All received frames with a multicast destination address (first bit in the destination address field is '1' and not all bits in the destination are '1') are passed
3	DAIFLT	Destination address inverse filtering bit This bit makes the result of DA filtering inverse. 0: Not inverse DA filtering result 1: Inverse DA filtering result
2	HMF	Hash multicast filter bit 0: The filter uses perfect mode for filtering multicast frame. 1: The filter uses hash mode for filtering multicast frame
1	HUF	Hash unicast filter bit 0: The filter uses perfect mode for filtering unicast frame 1: The filter uses hash mode for filtering unicast frame
0	PM	Promiscuous mode bit 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'. 0: Promiscuous mode disabled 1: Promiscuous mode enabled

32.4.3. 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															

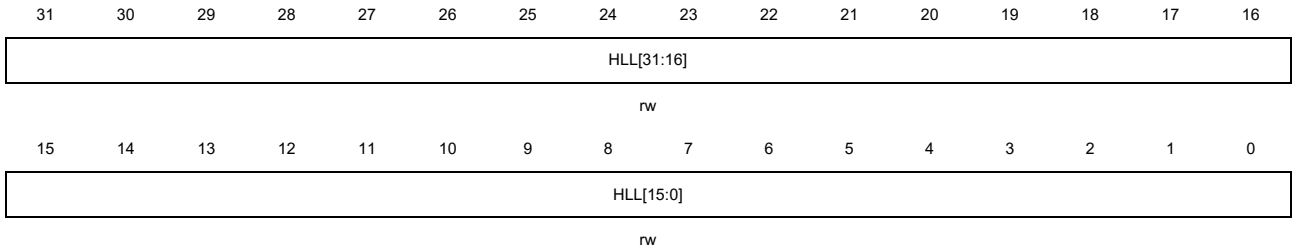
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HLH[15:0]															
rw															

Bits	Fields	Descriptions
31:0	HLH[31:0]	Hash list high bits These bits take the high 32-bit value of hash list.

32.4.4. 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).



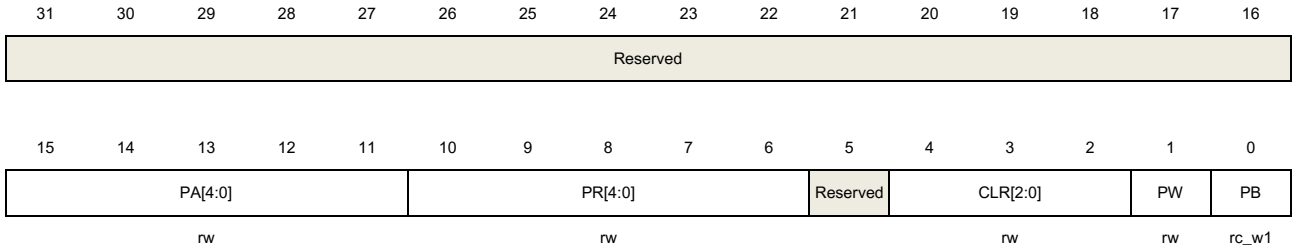
Bits	Fields	Descriptions
31:0	HLL[31:0]	Hash list low bits These bits take the low 32-bit value of hash list.

32.4.5. 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).



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:11	PA[4:0]	PHY address bits These bits choose which PHY device is to be accessed.
10:6	PR[4:0]	PHY register bits 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 MDC clock divided factor select which is decided by HCLK frequency range. 0x0: HCLK / 42 (HCLK range: 60-100 MHz) 0x1: HCLK / 62 (HCLK range: 100-150 MHz) 0x2: HCLK / 16 (HCLK range: 20-35 MHz) 0x3: HCLK / 26 (HCLK range: 35-60 MHz) 0x4: HCLK / 102 (HCLK range: 150-200 MHz) other: Reserved
1	PW	PHY write bit This bit indicates the PHY operation mode.

0: Sending read operation to PHY
1: Sending write operation to PHY

0	PB	PHY busy bit 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.

32.4.6. 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 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.

32.4.7. 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 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 0: Enable automatic zero-quanta generation function for pause control frame 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 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. 0x0: Pause time minus 4 slot times 0x1: Pause time minus 28 slot times 0x2: Pause time minus 144 slot times 0x3: Pause time minus 256 slot times **Note:** One slot time equals the time of transmitting 512 bits on the MII interface.
3	UPFDT	Unicast pause frame detect bit 0: Only the unique multicast address for pause frame which is specified in IEEE802.3 can be detected 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 0: Decode function for pause frame is disabled 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 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. 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 This bit only valid when TFCEN is set. This bit can send a pause frame in Full-duplex mode or activate the back pressure function in Half-duplex mode by application.

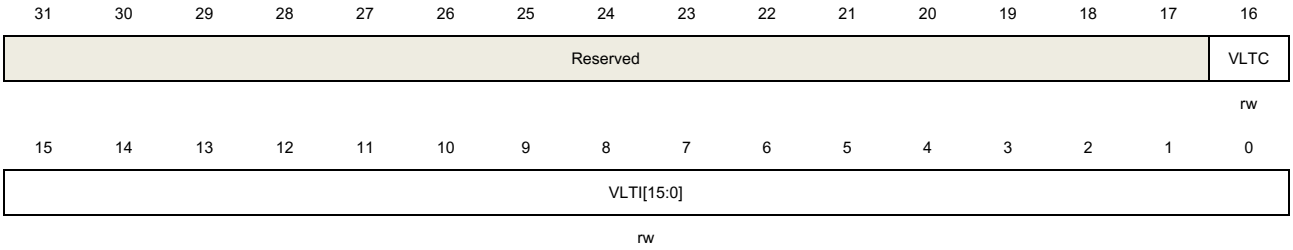
For Full-duplex mode, application must make sure this bit is 0 before writing ENET_MAC_FCTL register. After set by application, MAC sends a pause frame to interface and this bit will keep set until the pause frame has completed transmitting. For Half-duplex mode, MAC can enter back-pressure state by 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.

32.4.8. 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 This bit selects 12 or 16 bit VLAN tag for comparison. 0: All 16 bits (the 15th and 16th byte) of the VLAN tag in received frame are used for comparison 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 These bits are configured for detecting VLAN frame using 802.1Q VLAN tag format. The format shows below: VLTI[15:13]: UP (user priority) VLTI[12]: CFI (canonical format indicator) VLTI[11:0]: VID (VLAN identifier) 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.

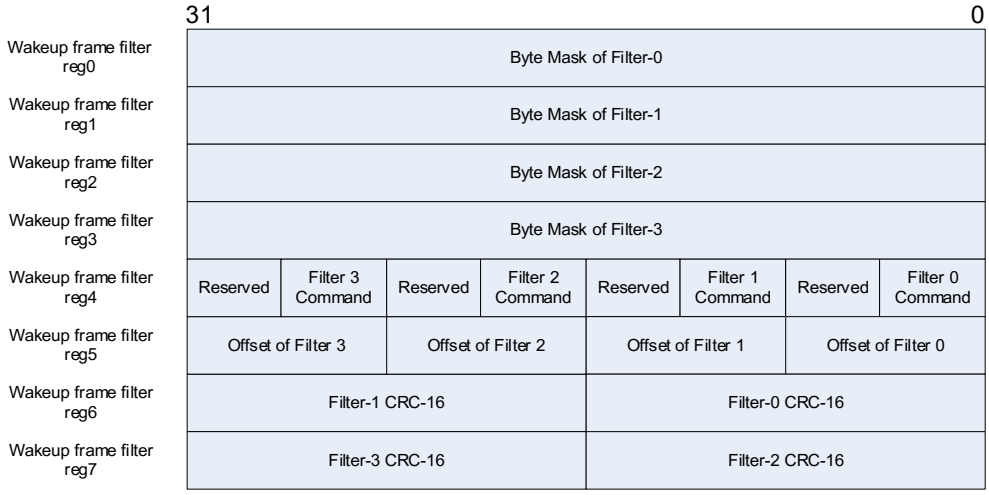
32.4.9. MAC remote wakeup frame filter register (ENET_MAC_RWFF)

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 32-15. Wakeup frame filter register

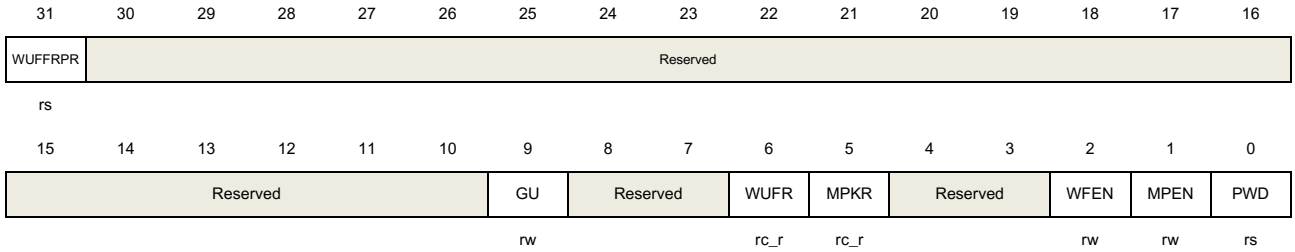


32.4.10. MAC wakeup management register (ENET_MAC_WUM)

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 This bit can reset the inner pointer of ENET_MAC_RWFF register by application set it to 1. Hardware clears it when resetting completes.

		0: No effect 1: Reset the ENET_MAC_RWFF register inner pointer
30:10	Reserved	Must be kept at reset value.
9	GU	Global unicast bit 0: Not all of received unicast frame is considered to be a wakeup frame 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 This bit is cleared when this register is read. 0: Has not received the wake-up frame 1:The wakeup event was generated due to reception of a wakeup frame
5	MPKR	Magic packet received bit This bit is cleared when this register is read. 0: Has not received the Magic Packet frame 1:The wakeup event was generated by the reception of a Magic Packet frame
4:3	Reserved	Must be kept at reset value.
2	WFEN	Wakeup frame enable bit 0: Disable generating a wakeup event due to wakeup frame reception 1: Enable generating a wakeup event due to wakeup frame reception
1	MPEN	Magic Packet enable bit 0: Disable generating a wakeup event due to Magic Packet reception 1: Enable generating a wakeup event due to Magic Packet reception
0	PWD	Power down bit 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.

32.4.11. 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 0: TxFIFO not full 1: TxFIFO is full
24	TXFNE	TxFIFO not empty flag 0: TxFIFO is empty 1: TxFIFO is not empty
23	Reserved	Must be kept at reset value.
22	TXFW	TxFIFO is writing 0: TxFIFO is not doing write operation 1: TxFIFO is doing write operation
21:20	TXFRS[1:0]	TxFIFO read operation status 0x0: Idle state 0x1: Reading state 0x2: Waiting feedback Tx status from MAC transmitter 0x3: Write Tx descriptor status or flush the TxFIFO
19	PCS	Pause condition status 0: MAC transmitter is not in pause condition 1: MAC transmitter is under pause condition and will delay transmitting frame
18:17	SOMT[1:0]	Status of MAC transmitter 0x0: Idle state 0x1: Waiting feedback of previous frame status or the end of IFG / BACKOFF period 0x2: For Full-duplex mode, indicates pause control frame is transmitting 0x3:Reading input frame from FIFO for transmission
16	MTNI	MAC transmit state not idle 0: MAC transmitter is in idle state 1: MAC transmitter is not in idle state
15:10	Reserved	Must be kept at reset value.
9:8	RXFS	RxFIFO state 0x0: RxFIFO empty 0x1: RxFIFO number of value is lower than flow-control low threshold 0x2: RxFIFO number of value is greater than flow-control high threshold 0x3: RxFIFO full
7	Reserved	Must be kept at reset value.
6:5	RXFRS[1:0]	RxFIFO read operation status 0x0: Idle state

		0x1: Reading state 0x2: Reading frame status (including time-stamp) 0x3: Flushing frame
4	RXFW	RxFIFO is writing 0: RxFIFO is not doing write operation 1: RxFIFO is doing write operation
3	Reserved	Must be kept at reset value.
2:1	RXAFS[1:0]	Rx asynchronous FIFO status RXAFS[1]:Rx asynchronous FIFO reading state in HCLK Clock domain RXAFS[0]:Rx asynchronous FIFO writing state in MAC RX_CLK Clock domain
0	MRNI	MAC receive state not idle 0: MAC receiver is in idle state 1: MAC receiver is not in idle state

32.4.12. 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 This bit is cleared when ENET_PTP_TSF register is read. 0: The system time value is less than the value specified in the target time registers 1: The system time value equals or exceeds the value specified in the target time registers
8:7	Reserved	Must be kept at reset value.
6	MSCT	MSC transmit status bit 0: All the bits in register ENET_MSC_TINTF are cleared 1: An interrupt is generated in the ENET_MSC_TINTF register
5	MSCR	MSC receive status bit 0: All the bits in register ENET_MSC_RINTF are cleared

		1: An interrupt is generated in the ENET_MSC_RINTF register
4	MSC	MSC status bit This bit is logic ORed from MSCT and MSCR bit. 0: Both MSCT and MSCR bits in this register are low 1: Any of bit 6 (MSCT) or bit 5 (MSCR) is set high
3	WUM	WUM status bit This bit is logic ORed from WUFR and MPKR bit in ENET_MAC_WUM register. 0: Wakeup frame or Magic Packet frame is not received 1: A Magic packet or remote wakeup frame is received in power down Mode
2:0	Reserved	Must be kept at reset value.

32.4.13. 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).

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						TMSTIM	Reserved					WUMIM	Reserved		
						rw						rw			

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	TMSTIM	Timestamp trigger interrupt mask bit 0: Unmask the timestamp interrupt generation 1: Mask the timestamp interrupt generation
8:4	Reserved	Must be kept at reset value.
3	WUMIM	WUM interrupt mask bit 0: Unmask the interrupt generation due to the WUM bit in ENET_MAC_INTF register 1: Mask the interrupt generation due to the WUM bit in ENET_MAC_INTF register
2:0	Reserved	Must be kept at reset value.

32.4.14. 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).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR0H[15:0]															
rw															

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 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.

32.4.15. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ADDR0L[31:16]														
	rw														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR0L[15:0]														
	rw														

Bits	Fields	Descriptions
31:0	ADDR0L[31:0]	MAC addresss0 low 32-bit 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.

32.4.16. 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 0: The address filter ignores the MAC address1 for filtering 1: The address filter uses the MAC address1 for perfect filtering
30	SAF	Source address filter bit 0: The MAC address1[47:0] is used to comparing with the DA field of the received frame 1: The MAC address1[47:0] is used to comparing with the SA field of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA / SA with the contents of the MAC address1 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR1H[15:8] MB[4]: ENET_MAC_ADDR1H[7:0] MB[3]: ENET_MAC_ADDR1L[31:24] MB[2]: ENET_MAC_ADDR1L[23:16] MB[1]: ENET_MAC_ADDR1L[15:8] 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 This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address1.

32.4.17. **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
ADDR1L[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR1L[15:0]															
rw															

Bits	Fields	Descriptions

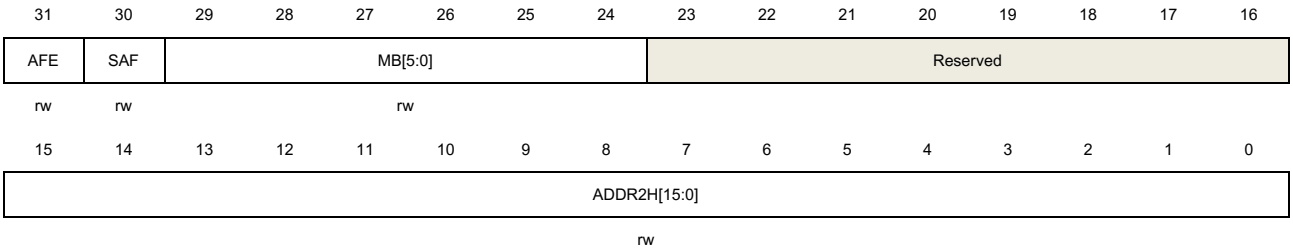
31:0	ADDR1L[31:0]	MAC address1 low 32-bit This field contains the low 32-bit of the 6-byte MAC address1.

32.4.18. 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 0: The address filter ignores the MAC address2 for filtering 1:The address filter uses the MAC address2 for perfect filtering
30	SAF	Source address filter bit 0: The MAC address2[47:0] is used to comparing with the DA fields of the received frame 1:The MAC address2[47:0] is used to comparing with the SA fields of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA / SA with the contents of the MAC address2 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR2H[15:8] MB[4]: ENET_MAC_ADDR2H[7:0] MB[3]: ENET_MAC_ADDR2L[31:24] MB[2]: ENET_MAC_ADDR2L[23:16] MB[1]: ENET_MAC_ADDR2L[15:8] 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 This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address2.

32.4.19. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR2L[31:16]															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR2L[15:0]															
rw															

Bits	Fields	Descriptions
31:0	ADDR2L[31:0]	MAC address2 low 32-bit This field contains the low 32-bit of the 6-byte MAC address2.

32.4.20. 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).

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
ADDR3H[15:0]															
rw															

Bits	Fields	Descriptions
31	AFE	Address filter enable bit 0: The address filter ignores the MAC address3 for filtering 1:The address filter use the MAC address3 for perfect filtering
30	SAF	Source address filter bit 0: The MAC address3[47:0] is used to comparing with the DA fields of the received frame 1:The MAC address3[47:0] is used to comparing with the SA fields of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA / SA with the contents of the MAC address3 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR3H[15:8] MB[4]: ENET_MAC_ADDR3H[7:0] MB[3]: ENET_MAC_ADDR3L[31:24]

MB[2]: ENET_MAC_ADDR3L[23:16]
MB[1]: ENET_MAC_ADDR3L[15:8]
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 This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address3.

32.4.21. 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 This field contains the low 32-bit of the 6-byte MAC address3.

32.4.22. 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 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.
0x0: 256 bytes
0x1: 512 bytes
0x2: 768 bytes
0x3: 1024 bytes
0x4: 1280 bytes
0x5: 1536 bytes
0x6,0x7: 1792 bytes

3	Reserved	Must be kept at reset value.
2:0	RFA[2:0]	Threshold of active flow control 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. 0x0: 256 bytes 0x1: 512 bytes 0x2: 768 bytes 0x3: 1024 bytes 0x4: 1280 bytes 0x5: 1536 bytes 0x6,0x7: 1792 bytes

32.4.23. 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).

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										AFHPM	PMC	MCFZ	RTOR	CTSR	CTR
										rw	wo	rw	rw	rw	rw

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

32.4.24. MSC receive interrupt flag register (ENET_MSC_RINTF)

Address offset: 0x0104
Reset value: 0x0000 0000

The interrupt flag bit of this register needs to be cleared by reading the corresponding MSC counter that triggered the interrupt.

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														RGUF	Reserved
														rc_r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									RFAE	RFCE	Reserved				
									rc_r	rc_r					

Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	RGUF	Received good unicast frames bit 0: Good unicast frame received counter is less than half of the maximum value 1: Good unicast frame received counter reaches half of the maximum value **Note**: This flag bit needs to be cleared by reading the ENET_MSC_RGUFCNT register

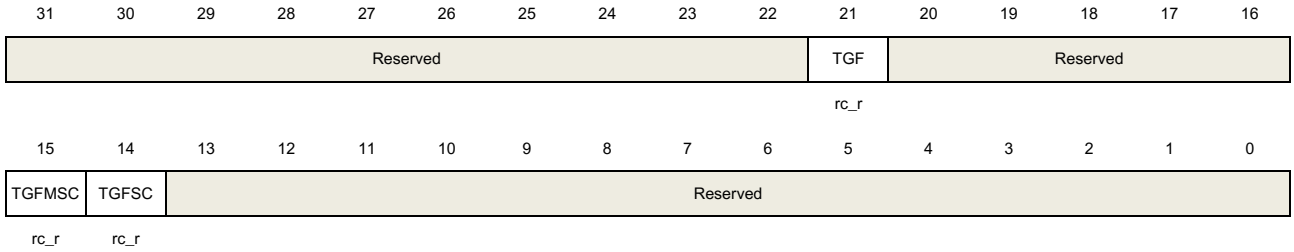
16:7	Reserved	Must be kept at reset value.
6	RFAE	Received frames alignment error bit 0: Alignment error frame received counter is less than half of the maximum value 1: Alignment error frame received counter reaches half of the maximum value、 **Note**: This flag bit needs to be cleared by reading the ENET_MSC_RFAECNT register
5	RFCE	Received frames CRC error bit 0: CRC error frame received counter is less than half of the maximum value 1: CRC error frame received counter reaches half of the maximum value **Note**: This flag bit needs to be cleared by reading the ENET_MSC_RFCECNT register
4:0	Reserved	Must be kept at reset value.

32.4.25. MSC transmit interrupt flag register (ENET_MSC_TINTF)

Address offset: 0x0108
Reset value: 0x0000 0000

The interrupt flag bit of this register needs to be cleared by reading the corresponding MSC counter that triggered the interrupt.

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 0: Good frame transmitted counter is less than half of the maximum value 1: Good frame transmitted counter reaches half of the maximum value **Note**: This flag bit needs to be cleared by reading the ENET_MSC_TGFCNT register
20:16	Reserved	Must be kept at reset value.
15	TGFMSC	Transmitted good frames more single collision bit 0: Good frame after more than a single collision transmitted counter is less than half of the maximum value 1:Good frame after more than a single collision transmitted counter reaches half of the maximum value

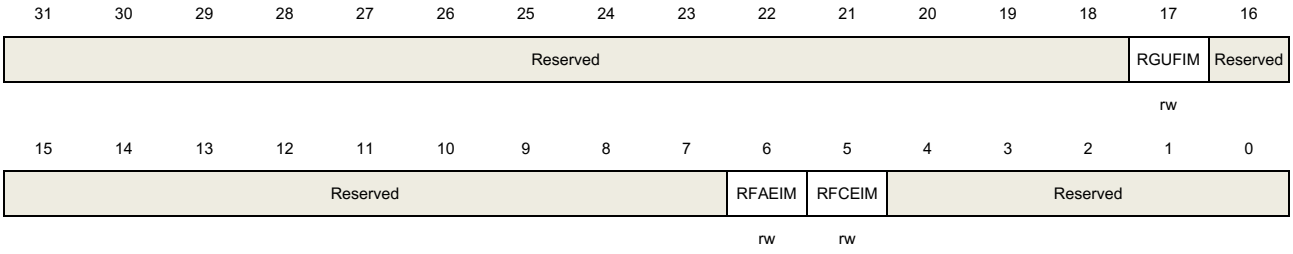
		Note: This flag bit needs to be cleared by reading the ENET_MSC_MSCCNT register
14	TGFSC	Transmitted good frames single collision bit 0: Good frame after a single collision transmitted counter is less than half of the maximum value 1: Good frame after a single collision transmitted counter reaches half of the maximum value **Note**: This flag bit needs to be cleared by reading the ENET_MSC_SCCNT register
13:0	Reserved	Must be kept at reset value.

32.4.26. 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 0: Unmask the interrupt when the RGUF bit is set 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 0: Unmask the interrupt when the RFAE bit is set 1: Mask the interrupt when the RFAE bit is set
5	RFCEIM	Received frame CRC error interrupt mask bit 0: Unmask the interrupt when RFCE bit is set 1: Mask the interrupt when the RFCE bit is set
4:0	Reserved	Must be kept at reset value.

32.4.27. 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

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFMSCIM	TGFSCIM	Reserved													
rw	rw														

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21	TGFIM	Transmitted good frames interrupt mask bit 0: Unmask the interrupt when the TGF bit is set 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 0: Unmask the interrupt when the TGFMSC bit is set 1: Mask the interrupt when the TGFMSC bit is set
14	TGFSCIM	Transmitted good frames single collision interrupt mask bit 0: Unmask the interrupt when the TFGSC bit is set 1: Mask the interrupt when the TFGSC bit is set
13:0	Reserved	Must be kept at reset value.

32.4.28. 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 These bits count the number of a transmitted good frames after only a single collision.

32.4.29. 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 These bits count the number of a transmitted good frames after more than one single collision.

32.4.30. 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

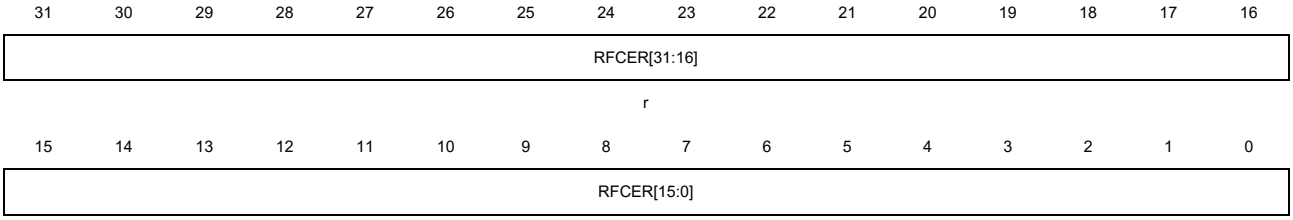
Bits	Fields	Descriptions
31:0	TGF[31:0]	Transmitted good frames counter bits These bits count the number of transmitted good frames.

32.4.31. 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.



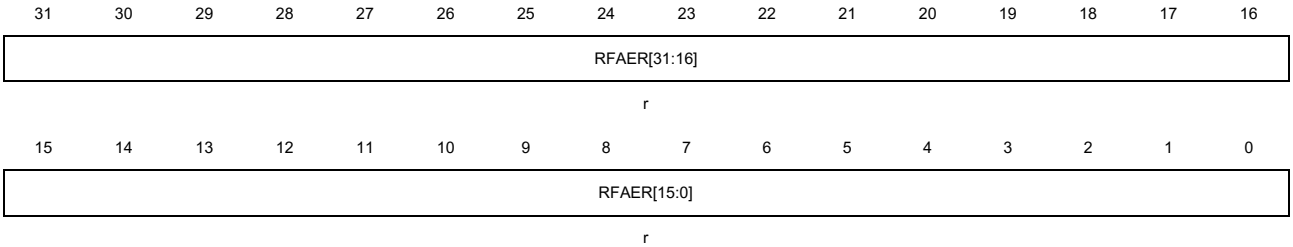
Bits	Fields	Descriptions
31:0	RFCER[31:0]	Received frames with CRC error counter bits These bits count the number of receive frames with CRC error.

32.4.32. 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.



Bits	Fields	Descriptions

31:0	RFAER[31:0]	Received frames alignment error counter bits These bits count the number of receive frames with alignment error.

32.4.33. 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 These bits count the number of good unicast frames received.

32.4.34. 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 0: No effect 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 0x0: Type of ordinary clock 0x1: Type of boundary clock 0x2: Type of end-to-end transparent clock 0x3: Type of peer-to-peer transparent clock
15	MNMSEN	Received master node message snapshot enable This bit is valid only when CKNT=0x0 or 0x1. 0: Snapshot is only taken for slave node message 1: Snapshot is only take for master node message
14	ETMSEN	Received event type message snapshot enable 0: All type messages are taken snapshot except Announce, Management and Signaling message 1: Only event type messages (SYNC, Delay_Req, Pdelay_Req and Pdelay_Resp) are taken snapshot
13	IP4SEN	Received PTP frame over IPv4-UDP snapshot enable 0: Do not take snapshot for PTP frame over IPv4-UDP 1: Take snapshot for PTP frame over IPv4-UDP
12	IP6SEN	Received PTP frame over IPv6-UDP snapshot enable 0: Do not take snapshot for PTP frame over IPv6-UDP 1: Take snapshot for PTP frame over IPv6-UDP
11	ESEN	Received PTP over Ethernet II packets snapshot enable 0: Do not take snapshot when received PTP over Ethernet II frame 1: Take snapshot when received PTP over Ethernet II frame
10	PFSV	PTP frame snooping version 0: Version 1 (Revision of IEEE STD. 1588-2002 / 1588-2008) 1: Version 2 (Revision of IEEE STD. 1588-2008)
9	SCROM	Subsecond counter rollover mode 0: Binary rollover mode. Subsecond rollovers when reach 0x7FFF_FFFF 1: Digital rollover mode. Subsecond rollovers when reach 0x3B9A_C9FF (0d999_999_999)
8	ARFSEN	All received frames snapshot enable 0: Not all received frames are taken snapshot 1: All received frames are taken snapshot
7:6	Reserved	Must be kept at reset value.
5	TMSARU	Time stamp addend register update bit This bit is cleared when the update is completed. This register bit must be read as zero before application set it. 0: The timestamp addend register's contents are not updated to the PTP block for

fine correction
1: The timestamp addend register's contents are updated to the PTP block for fine correction

4 TMSITEN Timestamp interrupt trigger enable bit
0: Disable timestamp interrupt
1: A timestamp interrupt is generated when the system time becomes greater than the value written in target time register
Note: When the timestamp trigger interrupt generated, this bit is cleared.

3 TMSSTU Timestamp system time update bit
Both the TMSSTU and TMSSTI bits must be read as zero before application set this bit.
0: The system time is maintained without any change
1: The system time is updated (added to or subtracted from) with the value specified in the timestamp update (high and low) registers. It is cleared by hardware when the update finished.

2 TMSSTI Timestamp system time initialize bit
This bit must be read as zero before application set it.
0: The system time is maintained without any change
1: Initializing the system time with the value in timestamp update (high and low) registers. It is cleared by hardware when the initialization finished.

1 TMSFCU Timestamp fine or coarse update bit
0: The system timestamp uses the coarse method for updating
1:The system timestamp uses the fine method for updating

0 TMSEN Timestamp enable bit
0: Disable timestamp function
1: Enable timestamp function for transmit and receive frames
Note: After setting this to 1, application must initialize the system time.

Table 32-7. Supported time stamp snapshot with PTP register configuration

CKNT (Bit 17:16)	MNMSEN (Bit 15)	ETMSEN (Bit 14)	Supported message type for snapshot
0X	X(*)	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
0X	1	1	Delay_Req
0X	0	1	SYNC
10	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
10	X	1	SYNC, Follow_Up
11	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp
11	X	1	SYNC, Pdelay_Req, Pdelay_Resp

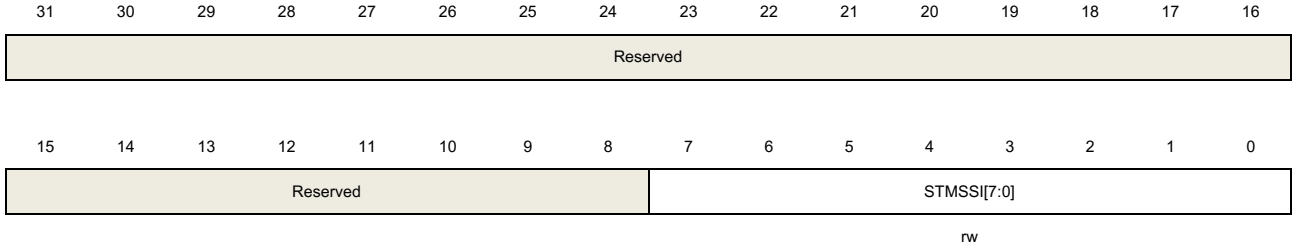
*: X means do not care

32.4.35. 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.

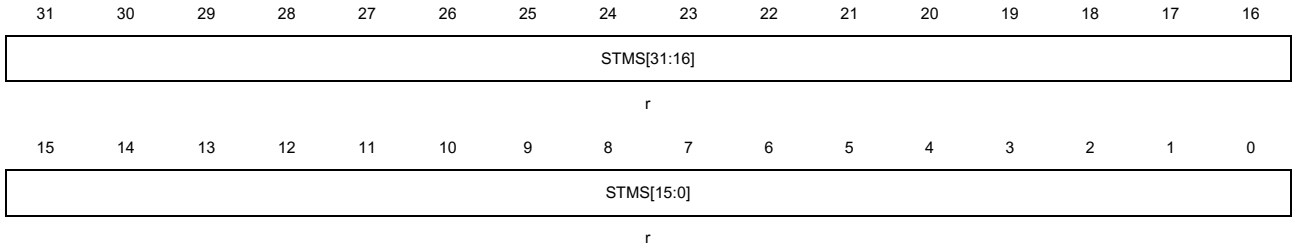


Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	STMSSI[7:0]	System time subsecond increment bits In every update operation, these bits are added to the subsecond value of system time.

32.4.36. 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).



Bits	Fields	Descriptions
31:0	STMS[31:0]	System time second bits These bits show the current second of the system time.

32.4.37. 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).

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

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

Bits	Fields	Descriptions
31	STS	System time sign bit 0: Time value is positive 1: Time value is negative
30:0	STMSS[30:0]	System time subseconds bits These bits show the current subsecond of the system time with 0.46 ns accuracy if required accuracy is 20 ns.

32.4.38. 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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMSUS[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	TMSUS[31:0]	Time stamp update second bits These bits are used for initializing or adding / subtracting to second of the system time.

32.4.39. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMSUPNS	TMSUSS[30:16]														
rw	rw														

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

Bits	Fields	Descriptions
31	TMSUPNS	Timestamp update positive or negative sign bit When TMSSTI is set, this bit must be 0. 0: Timestamp update value is added to system time 1: Timestamp update value is subtracted from system time
30:0	TMSUSS[30:0]	Timestamp update subsecond bits These bits are used for initializing or adding / subtracting to subsecond of the system time.

32.4.40. 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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMSA[31:16]															
rw															

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

Bits	Fields	Descriptions
31:0	TMSA[31:0]	Time stamp addend bits These registers contain a 32-bit time value which is added to the accumulator register to achieve time synchronization.

32.4.41. 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 These bits store the expected target second time.

32.4.42. 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 These bits store the expected target nanosecond time (signed).

32.4.43. 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 0: System time is not equal or greater than target time. 1: System time is equal or greater than target time **Note:** Reading ENET_PTP_TSF register will clear this bit.
0	TSSCO	Timestamp second counter overflow bit 0: Timestamp second counter has not overflowed 1: Timestamp second counter is greater than 0xFFFF FFFF

32.4.44. 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).

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												PPSOFC[3:0]			
												rw			

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3:0	PPSOFC	PPS output frequency configure 0x0: 1Hz (Pulse width: 125ms for binary rollover, 100ms for digital rollover) 0x1: 2Hz (Pulse width: 50% duty cycle for binary rollover) 0x2: 4Hz (Pulse width: 50% duty cycle for binary rollover) …. 0xF: 32768 (2^{15}) Hz (Pulse width: 50% duty cycle for binary rollover) **Note:** If digital rollover is selected, only PPSOFC=0 is recommended.

32.4.45. 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MB	AA	FPBL	UIP	RXDP[5:0]						FB
					rw	rw	rw	rw	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RTPR[1:0]	PGBL[5:0]	DFM	DPSL[4:0]	DAB	SWR
rw	rw	rw	rw	rw	rs

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	MB	Mixed burst 0: AHB master interface only transfer fixed burst length with 16 and below 1: AHB master interface will transfer burst length greater than 16 with INCR **Note:** MB and FB should be and must be only one of bit is set.
25	AA	Address-aligned bit 0: Disable address-aligned 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 0: The PGBL value programmed (bits[22:17] and bits[13:8]) for the DMA data number of beats to be transferred 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 0: The PGBL value in bits[13:8] is applicable for both TxDMA and RxDMA engines 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 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. 0x01: max beat number is 1 0x02: max beat number is 2 0x04: max beat number is 4 0x08: max beat number is 8 0x10: max beat number is 16 0x20: max beat number is 32 Other: Reserved
16	FB	Fixed burst bit 0: The AHB can use SINGLE and INCR burst transfer operations 1: The AHB can only use SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers **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 These bits indicate the access ratio between RxDMA and TxDMA.

0x0: RxDMA : TxDMA = 1:1
0x1: RxDMA : TxDMA = 2:1
0x2: RxDMA : TxDMA = 3:1
0x3: RxDMA : TxDMA = 4:1
Note: This bit is valid only when the arbitration mode is Round-robin (DAB=0).

13:8	PGBL[5:0]	Programmable burst length bits 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. 0x01: max beat number is 1 0x02: max beat number is 2 0x04: max beat number is 4 0x08: max beat number is 8 0x10: max beat number is 16 0x20: max beat number is 32 Other: Reserved.
7	DFM	Descriptor format mode 0: Normal mode descriptor 1: Enhanced mode descriptor
6:2	DPSL[4:0]	Descriptor skip length bit These bits are valid only between two ring mode descriptors. They define the number of words (32-bit) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When DPSL value equals zero, the descriptor table is taken as contiguous by the DMA, in ring mode.
1	DAB	DMA arbitration bit This bit indicates the arbitration mode between RxDMA and TxDMA. 0: Round-robin mode and DMA access priority is given in RTPR 1: Fixed mode. RxDMA has higher priority than TxDMA
0	SWR	Software reset bit This bit can reset all core internal registers located in CLK_TX and CLK_RX. It is cleared by hardware when the reset operation is complete in all clock domains. **Note:** Application must make sure this bit is 0 before writing any MAC core registers. 0: Core and inner register are not in reset state 1: Reset all core internal registers

32.4.46. 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 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.

32.4.47. 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 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.

32.4.48. 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 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.

32.4.49. 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 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.

32.4.50. 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 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. 0: Timestamp event has not occurred 1: Timestamp event has occurred
28	WUM	WUM status bit 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. 0: WUM event has not occurred 1: WUM event has occurred
27	MSC	MSC status bit 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. 0: MSC event has not occurred 1: MSC event has occurred
26	Reserved	Must be kept at reset value.
25:23	EB[2:0]	Error bits status bit When FBE=1, these bits decode the type of error that caused a bus response error on AHB bus. EB[0] 1: Error during data transfer by TxDMA. 0: Error during data transfer by RxDMA

		EB[1] 1: Error during read transfer. 0: Error during write transfer EB[2] 1: Error during descriptor access. 0: Error during data buffer access
22:20	TP[2:0]	Transmit process state bit These bits decode the TxDMA state. 0x0: Stopped; Reset or Stop Transmit Command issued 0x1: Running; Fetching transmit transfer descriptor 0x2: Running; Waiting for status 0x3: Running; Reading Data from host memory buffer and queuing it to transmit buffer (TxFIFO) 0x4, 0x5: Reserved 0x6: Suspended; Transmit descriptor unavailable or transmit buffer underflow 0x7: Running; Closing transmit descriptor
19:17	RP[2:0]	Receive process state bit These bits decode the RxDMA state. 0x0: Stopped: Reset or Stop Receive Command issued 0x1: Running: Fetching receive transfer descriptor 0x2: Reserved 0x3: Running: Waiting for receive packet 0x4: Suspended: Receive descriptor unavailable 0x5: Running: Closing receive descriptor 0x6: Reserved 0x7: Running: Transferring the receive packet data from receive buffer to host memory
16	NI	Normal interrupt summary The NI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register: TS (ENET_DMA_STAT[0]): Transmit interrupt TBU (ENET_DMA_STAT[2]): Transmit buffer unavailable RS (ENET_DMA_STAT[6]): Receive interrupt ER (ENET_DMA_STAT[14]): Early receive interrupt **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 The AI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register: TPS (ENET_DMA_STAT[1]): Transmit process stopped TJT (ENET_DMA_STAT[3]): Transmit jabber timeout RO (ENET_DMA_STAT[4]): Receive FIFO overflow TU (ENET_DMA_STAT[5]): Transmit underflow RBU (ENET_DMA_STAT[7]): Receive buffer unavailable

		RPS (ENET_DMA_STAT[8]): Receive process stopped RWT (ENET_DMA_STAT[9]): Receive watchdog timeout ET (ENET_DMA_STAT[10]): Early transmit interrupt FBE (ENET_DMA_STAT[13]): Fatal bus error **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 This bit is automatically cleared when the ENET_DMA_STAT[6] is set. 0: The first buffer has not been filled 1: The first buffer has filled with received frame
13	FBE	Fatal bus error status bit This bit indicates a response error on AHB interface is occurred and the error type can be decoded by EB[2:0] bits. 0: Bus error has not occurred 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 0: The frame to be transmitted has not fully transferred into the TxFIFO 1: The frame to be transmitted has fully transferred into the TxFIFO
9	RWT	Receive watchdog timeout status bit 0: No received a frame with a length greater than 2048 bytes 1: A frame with a length greater than 2048 bytes is received
8	RPS	Receive process stopped status bit 0: The receive process is not in stop state 1: The receive process is in stop state
7	RBU	Receive buffer unavailable status bit 0: The DAV bit in fetched next receive descriptor is set 1: The DAV bit in fetched next receive descriptor is reset and RxDMA enters suspend state
6	RS	Receive status bit 0: Frame reception has not completed 1: Frame reception has completed
5	TU	Transmit underflow status bit 0: Underflow error has not occurred during frame transmission 1: The TxFIFO encountered an underflow error during frame transmission and entered suspend state
4	RO	Receive overflow status bit 0: Receive overflow error has not occurred during frame reception 1: The RxFIFO encountered an overflow error during frame reception. If a part of

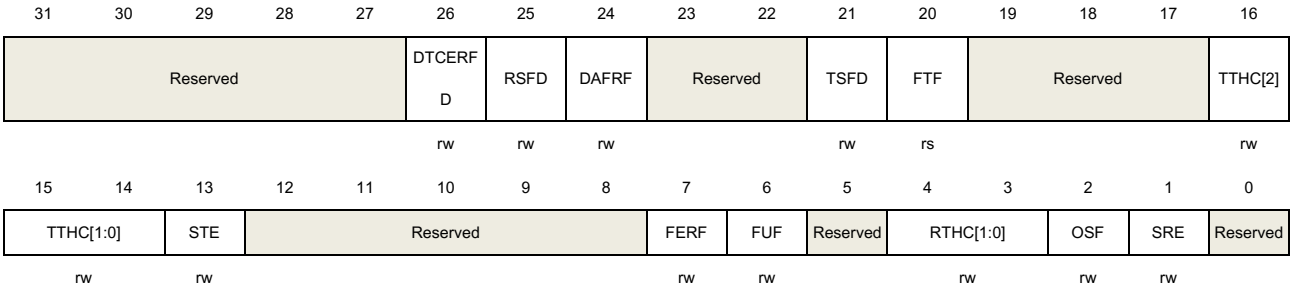
		frame data has transferred to the memory, the overflow status in RDES0[11] is also set.
3	TJT	Transmit jabber timeout status bit 0: Transmit jabber timeout has not occurred during frame transmission 1: The transmit jabber timer expired. The TxDMA controller cancels the current transmission and enters stop state. This also causes JT bit in TDES0 set.
2	TBU	Transmit buffer unavailable status bit 0: The DAV bit in fetched next transmit descriptor is set 1: The DAV bit in fetched next transmit descriptor is reset and TxDMA enters suspend state
1	TPS	Transmit process stopped status bit 0: The transmission is not in stop state 1: The transmission is in stop state
0	TS	Transmit status bit This bit can only be set when both LSG and INTC are set in TDES0. 0: Current frame transmission is not finished 1: Current frame transmission is finished.

32.4.51. 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.



Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value.
26	DTCERFD	Dropping of TCP / IP checksum error frames disable bit 0: All error frames will be dropped when FERF=0 1: The received frame with only payload error but no other errors will not be dropped.
25	RSFD	Receive Store-and-Forward bit 0: The RxFIFO operates in Cut-Through mode. The forwarding threshold depends

		on the RTHC bits 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 0: The RxDMA flushes all frames because of unavailable receive descriptor 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 0: The TxFIFO operates in Cut-Through mode. The TTHC bits in ENET_DMA_CTL register defines the start popping time from TxFIFO. 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. **Note:** This bit can be changed when transmission is in stop state.
20	FTF	Flush transmit FIFO bit 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. **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 These bits control the start transmitting byte threshold of the TxFIFO. When TSFD=1, these bits are ignored. 0x0: 64 0x1: 128 0x2: 192 0x3: 256 0x4: 40 0x5: 32 0x6: 24 0x7: 16
13	STE	Start / stop transmission enable bit 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. 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 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. 1: All frame received with error except runt error are forwarded to memory
6	FUF	Forward undersized good frames bit 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. 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 These bits control the threshold bytes of the RxFIFO. **Note:** These bits are valid only when the RSFD=0 and are ignored when the RSFD=1. 0x0: 64 0x1: 32 0x2: 96 0x3: 128
2	OSF	Operate on second frame bit 0: The TxDMA controller process the second transmit frame after the status of the first frame is written back to descriptor 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.

32.4.52. 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 0: A normal interrupt is disabled. 1: A normal interrupt is enabled This bit enables the following bits: TS (ENET_DMA_STAT[0]): Transmit interrupt. TBU (ENET_DMA_STAT[2]): Transmit buffer unavailable. RS (ENET_DMA_STAT[6]): Receive interrupt. ER (ENET_DMA_STAT[14]): Early receive interrupt.
15	AIE	Abnormal interrupt summary enable bit 0: An abnormal interrupt is disabled. 1: An abnormal interrupt is enabled This bit enables the following bits: TPS (ENET_DMA_STAT[1]): Transmit process stopped. TJT (ENET_DMA_STAT[3]): Transmit jabber timeout. RO (ENET_DMA_STAT[4]): Receive FIFO overflow. TU (ENET_DMA_STAT[5]): Transmit underflow. RBU (ENET_DMA_STAT[7]): Receive buffer unavailable.

		RPS (ENET_DMA_STAT[8]): Receive process stopped. RWT (ENET_DMA_STAT[9]): Receive watchdog timeout. ET (ENET_DMA_STAT[10]): Early transmit interrupt. FBE (ENET_DMA_STAT[13]): Fatal bus error.
14	ERIE	Early receive interrupt enable bit 0: The early receive interrupt is disabled 1: The early receive interrupt is enabled
13	FBEIE	Fatal bus error interrupt enable bit 0: The fatal bus error enable interrupt is disabled 1: The fatal bus error enable interrupt is enabled
12:11	Reserved	Must be kept at reset value.
10	ETIE	Early transmit interrupt enable bit 0: The early transmit interrupt is disabled 1: The early transmit interrupt is enabled
9	RWTIE	Receive watchdog timeout interrupt enable bit 0: The receive watchdog timeout interrupt is disabled 1: The receive watchdog timeout interrupt is enabled
8	RPSIE	Receive process stopped interrupt enable bit 0: The receive stopped interrupt is disabled 1: The receive stopped interrupt is enabled
7	RBUIE	Receive buffer unavailable interrupt enable bit 0: The receive buffer unavailable interrupt is disabled 1: The receive buffer unavailable interrupt is enabled
6	RIE	Receive interrupt enable bit 0: The receive interrupt is disabled 1: The receive interrupt is disabled
5	TUIE	Transmit underflow interrupt enable bit 0: The underflow interrupt is disabled 1: The underflow interrupt is enabled
4	ROIE	Receive overflow interrupt enable bit 0: The overflow interrupt is disabled 1: The overflow interrupt is enabled
3	TJTIE	Transmit jabber timeout interrupt enable bit 0: The transmit jabber timeout interrupt is disabled 1: The transmit jabber timeout interrupt is enabled
2	TBUIE	Transmit buffer unavailable interrupt enable bit 0: The transmit buffer unavailable interrupt is disabled 1: The transmit buffer unavailable interrupt is enabled

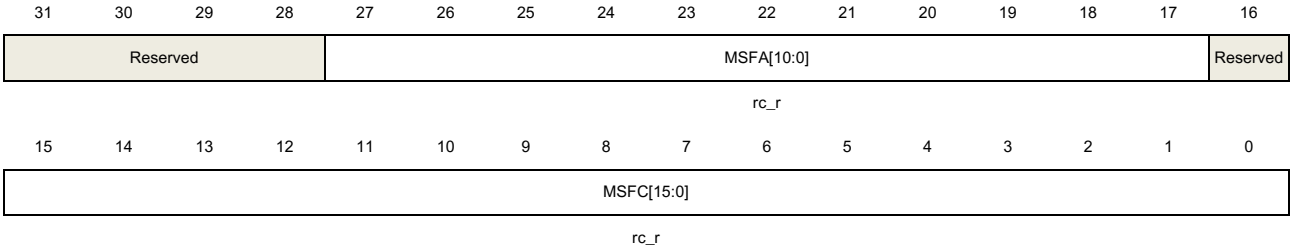
1	TPSIE	Transmit process stopped interrupt enable bit 0: The transmission stopped interrupt is disabled 1: The transmission stopped interrupt is enabled
0	TIE	Transmit interrupt enable bit 0: The transmit interrupt is disabled 1: The transmit interrupt is enabled

32.4.53. 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.



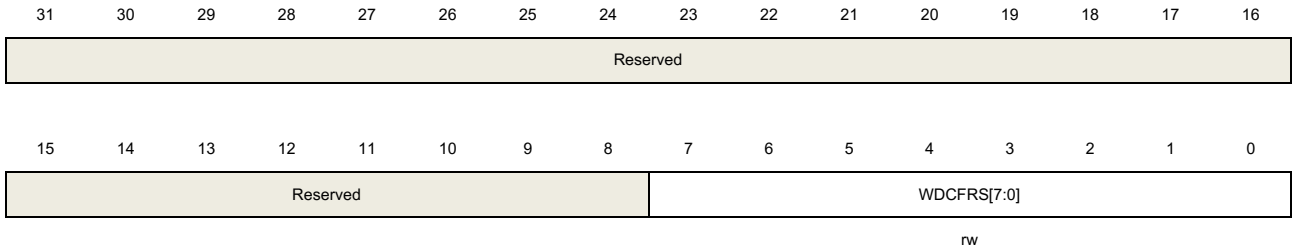
Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:17	MSFA[10:0]	Missed frames by the application bits These bits indicate the number of frames dropped by RxFIFO.
16	Reserved	Must be kept at reset value.
15:0	MSFC[15:0]	Missed frames by the controller bits These bits indicate the 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.

32.4.54. 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.



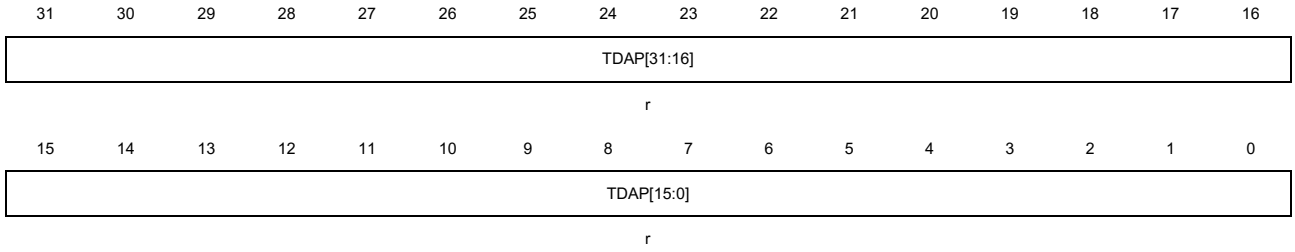
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	WDCFRS[7:0]	Watchdog counter for receive status (RS) bit These bits are only valid when DINTC (RXDES1) is set. When DINTC=1 and a frame is received, the RS bit will be set delay a time of WDCFRS*256 HCLK after receiving complete.

32.4.55. 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.



Bits	Fields	Descriptions
31:0	TDAP[31:0]	Transmit descriptor address pointer bits These bits are automatically updated by TxDMA controller during operation.

32.4.56. 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 These bits are automatically updated by RxDMA controller during operation.

32.4.57. 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 These bits are automatically updated by TxDMA controller during operation.

32.4.58. 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 These bits are automatically updated by RxDMA controller during operation.

33. Universal serial bus full-speed interface (USBFS)

33.1. Overview

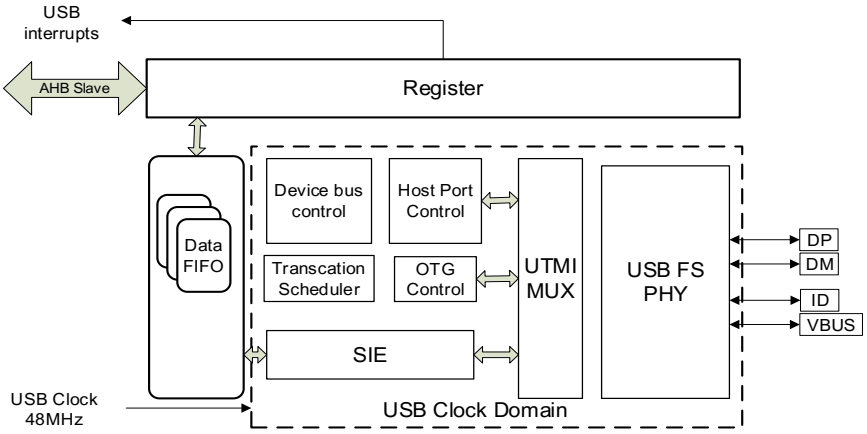
USB Full-Speed (USBFS) controller provides a USB-connection solution for portable devices. USBFS supports host and device modes, as well as OTG mode with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol). USBFS contains a full-speed internal USB PHY and external PHY chip is not contained. USBFS supports all the four types of transfer (control, bulk, Interrupt and isochronous) which defined in USB 2.0 protocol.

33.2. Characteristics

- Supports USB 2.0 host mode at Full-Speed(12Mb/s) or Low-Speed(1.5Mb/s)
- Supports USB 2.0 device mode at 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
- Includes a USB transaction scheduler in host mode to handle USB transaction request efficiently.
- Includes a 1.25KB FIFO RAM.
- Supports 8 channels in host mode.
- Includes 2 transmit FIFOs (periodic and non-periodic) and a receive FIFO (shared by all channels) in host mode.
- Includes 4 transmit FIFOs (one for each IN endpoint) and a receive FIFO (shared by all OUT endpoints) in device mode.
- Supports 4 OUT and 4 IN endpoints in device mode.
- Supports remote wakeup in device mode.
- Includes a Full-Speed USB PHY with OTG protocol supported.
- Time intervals of SOFs is dynamic adjustable in host mode.
- SOF pulse supports output to pad.
- Supports detecting ID pin level and VBUS voltage.
- Needs external component to supply power for connected USB device in host mode or OTG A-device mode.

33.3. Block diagram

Figure 33-1. USBFS block diagram



33.4. Signal description

Table 33-1. USBFS signal description

I/O port	Type	Description
VBUS	Input	Bus power port
DM	Input/Output	Differential D- port
DP	Input/Output	Differential D+ port
ID	Input	USB identification: Mini connector identification port

33.5. Function overview

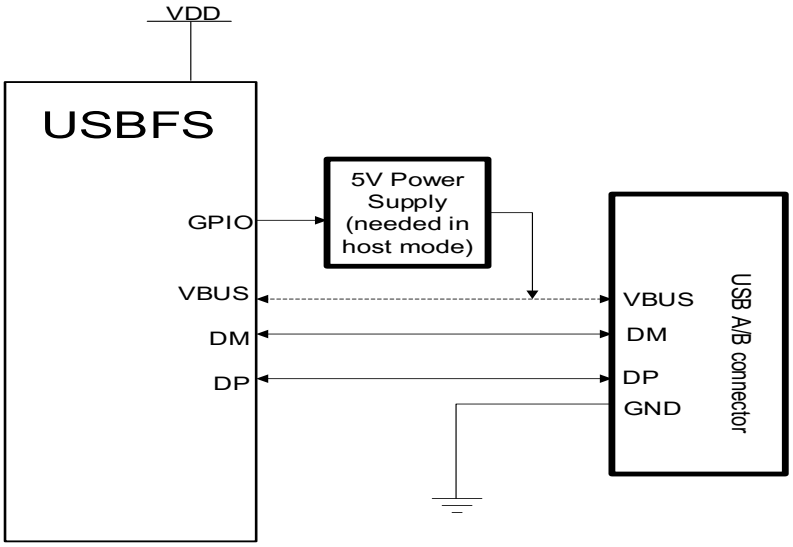
33.5.1. USBFS clocks and working modes

USBFS can operate as a host, a device or a DRD (Dual Role Device), it contains an internal full-speed PHY.The maximum speed supported by USBFS is full-speed.

The internal PHY supports Full-Speed and Low-Speed in host mode, supports Full-speed in device mode, and supports OTG mode with HNP and SRP. The USB clock used by the USBFS should be 48MHz. The 48MHz USB clock is generated from internal clocks in system, and its source and divider factors are configurable in RCU.

The pull-up and pull-down resistors have already been integrated into the internal PHY and they could be controlled by USBFS automatically according to the current mode (host, device or OTG mode) and connection status. A typical connection is shown in ***Figure 33-2. Connection with host or device mode***

Figure 33-2. Connection with host or device mode

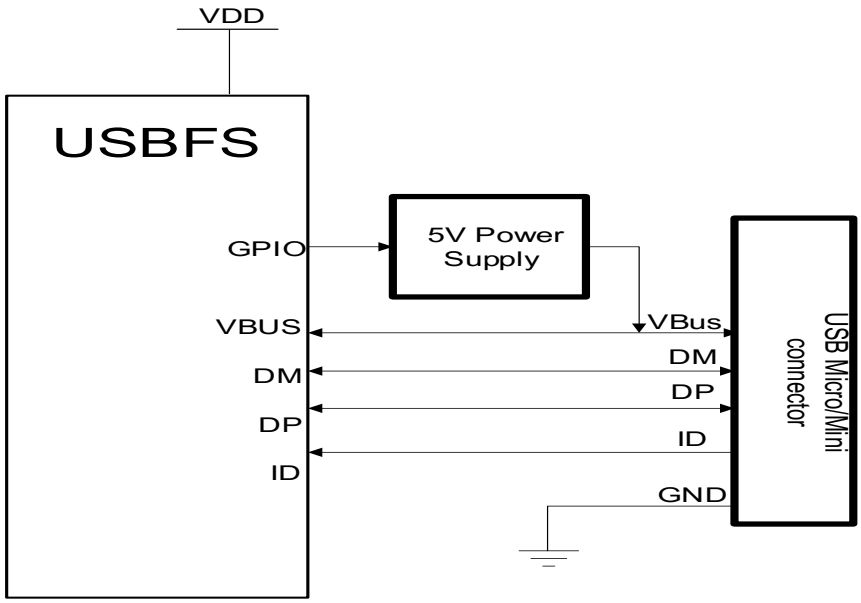


When USBFS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power detecting pin used for voltage detection defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuits on VBUS line. So if application needs VBUS power, an external power supply IC is needed. The VBUS connection between USBFS and the USB connector can be omitted in host mode, so USBFS doesn't detect the voltage level on VBUS pin and always assumes that the 5V power is present.

When USBFS works in device mode (FHM bit is cleared and FDM bit is set), the VBUS detection circuit is connected to a GPIO pin. USBFS continuously monitor the VBUS voltage by the GPIO pin and will immediately switch on the pull-up resistor on DP line once that the VBUS voltage rise above the needed valid value. This will cause a connection. If the VBUS voltage falls below the needed valid value, the pull-up resistor on DP line will be switched off and a disconnection will happen.

The OTG mode connection is described in the ***Figure 33-3. Connection with OTG mode***. When USBFS works in OTG mode, the FHM, FDM bits in USBFS_GUSBCS and VBUSIG bit in USBFS_GCCFG should be cleared. In this mode, the USBFS needs all the four pins: DM, DP, VBUS and ID, and needs to use several voltage comparers to monitor the voltage on these pins. USBFS also contains VBUS charge and discharge circuits to perform SRP request described in OTG protocol. The OTG A-device or B-device is decided by the level of ID pins. USBFS controls the pull-up or pull-down resistor during performing the HNP protocol.

Figure 33-3. Connection with OTG mode

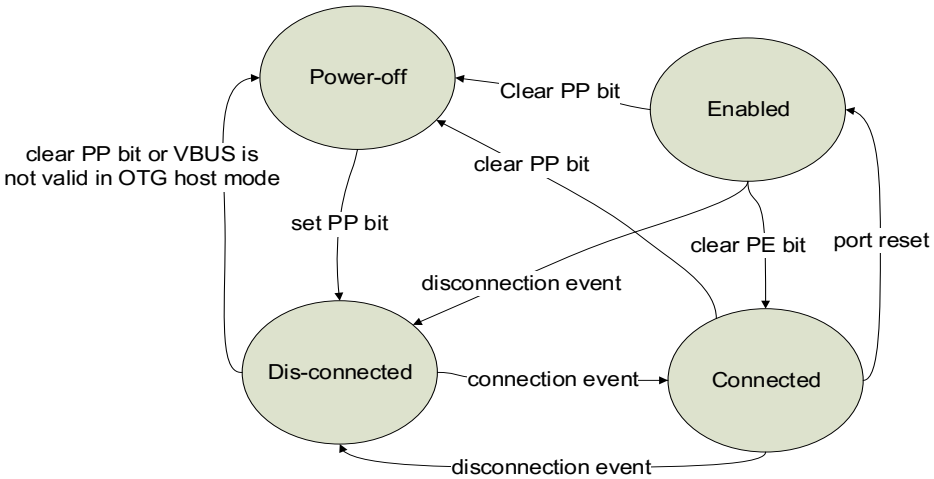


33.5.2. USB host function

USB Host Port State

Host application may control state of the USB port via USBFS_HPCS register. After system initialization, the USB port stays at power-off state. After PP bit is set by software, the internal USB PHY 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 33-4. State transition diagram of host port



Connection, Reset and Speed identification

As a USB host, USBFS will trigger a connection flag for application after a connection is

detected and will trigger a 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 connected or enabled state.

The USBFS performs speed identification during connection, and the speed information will be reported in PS filed in USBFS_HPCS register. USBFS identifies the device speed by the voltage level of DM or DP. As described in USB protocol, full-speed device pulls up DP line while low-speed device pulls up DM line.

Suspend and resume

USBFS supports suspend state and resume operation. When USBFS port is at enabled state, writing 1 to PSP bit in USBFS_HPCS register will cause USBFS to enter suspend state. In suspend state, USBFS 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 USBFS_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 USBFS_GINTF will be set and the USBFS wake up interrupt will be triggered if a host in suspend state detects a remote wakeup signal.

SOF generate

USBFS 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 in full-speed links.

Each time after USBFS enters into enabled state, it will send the SOF packet periodically which the time is defined in USB 2.0 protocol. In addition, application may adjust the length of a frame by writing FRI filed in USBFS_HFT registers. The FRI bits define the number of USB clock cycles in a frame, so its value should be calculated based on the frequency of USB clock which is used by USBFS. The FRT filed bits show that the remaining clock cycles of the current frame and stop changing during suspend state.

USBFS 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 USBFS_GCCFG register and configure the related pin registers in GPIO.

USB Channels and Transactions

USBFS includes 8 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 are all configured in channel related registers such as USBFS_HCHxCTL and USBFS_HCHxLEN.

USBFS supports all the four kinds of transfer types: control, bulk, interrupts and isochronous. USB 2.0 protocol divides these transfers into 2 kinds: non-periodic transfer (control and bulk) and periodic transfer (interrupt and isochronous). Based on this, USBFS includes two request queues: periodic request queue and non-periodic request queue, 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.

Application needs to write packet into data FIFO via AHB register interface if it wants to start an OUT transaction on USB bus. USBFS hardware will automatically generate a transaction request entry in request queue after the application writes a whole packet.

The request entries in request queue are processed in order by transaction control module. USBFS always tries to process periodic request queue firstly and secondly process non-periodic request queue.

After a start of frame, USBFS 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. Each time the USBFS reads and pops a request entry from request queue. If this is a channel disable request, it immediately disables the channel and prepares to process the next entry.

If the current request is a transaction request and the USB bus time is enough for this transaction, USBFS will employ SIE to generate this transaction on USB bus.

When the required bus time for the current request is not enough in the current frame, and if this is a periodic request, USBFS stops processing the periodic queue and starts to process non-periodic request. If this is a non-periodic queue the USBFS will stop processing any queue and wait until the end of current frame.

33.5.3. USB device function

USB Device Connection

In device mode, USBFS stays at power-off state after initialization. After connecting to a USB host with 5V power supply through VBUS pin, USBFS enters into powered state. USBFS begins to switch on the pull-up resistor on DP line and thus, host side will detect a connection event.

Note: The VBUS pin must be connected to the PA9 for detecting the level.

Reset and Speed-Identification

The USB host always starts a USB reset when it detects a device connection, and USBFS in device mode will trigger a reset interrupt by hardware when it detects the reset event on USB bus.

After reset sequence, USBFS will trigger an ENUMF interrupt in USBFS_GINTF register and reports current enumerated device speed in ES bits in USBFS_DSTAT register, this bit field is always 0b'11'(full-speed).

As required by USB 2.0 protocol, USBFS doesn't support low-speed in device mode.

Suspend and Wake-up

A USB device will enter into suspend state when the USB bus stays at IDLE state and there

is no change on data lines for 3ms. When USB device is in suspend state, most of its clock are closed to save power. The USB host is able to wake up the suspended device by generating a resume signal on USB bus. When USBFS detects the resume signal, the WKUPIF flag in USBFS_GINTF register will be set and the USBFS wake up interrupt will be triggered.

In suspend mode, USBFS is also able to remotely wake up the USB bus. Software may set RWKUP bit in USBFS_DCTL register to send 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

USBFS supports soft disconnection. After the device is powered on, USBFS will switch on the pull-up resistor on DP line so that the host can detect the connection. It is able to force a disconnection by setting the SD bit in USBFS_DCTL register. After the SD bit is set, USBFS will directly switch off the pull-up resistor, so that USB host will detect a disconnection on USB bus.

SOF tracking

When USBFS receives a SOF packet on USB bus, it will trigger a SOF interrupt and begin to count the bus time using local USB clock. The frame number of the current frame is reported in FNRSOF filed in USBFS_DSTAT register. When the USB bus time reaches EOF1 or EOF2 point (End of Frame, described in USB 2.0 protocol), USBFS will trigger an EOPFIF interrupt in USBFS_GINTF register. These flags and registers can be used to get current bus time and position information.

33.5.4. OTG function overview

USBFS supports OTG function described in OTG protocol 1.3, 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 to 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 being a Standard-A plug. The B-Device is a peripheral at the start of a session. USBFS uses the voltage level of ID pin to identify A-Device or B-Device. The ID status is reported in IDPS bit in USBFS_GOTGCS register. For the details of transfer states between A-Device and B-Device, please refer to OTG 1.3 protocol.

HNP

The Host Negotiation Protocol (HNP) allows the host function to be switched between two directly connected On-The-Go devices and eliminates the necessity of switching the cable connections for the change of control of communications between the devices. HNP will be initialized typically 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 a host or a device, depending that which type of plug (Micro-A plug for host, Micro-B plug for device) is inserted. By utilizing the Host Negotiation Protocol (HNP), an On-The-Go B-Device, which is the default device, may make a request to be a host. The process for the exchange of the role to a host is described in this section. This protocol eliminates the necessity of switching the cable connection for the change of the roles of the connected devices.

When USBFS is in OTG A-Device host mode and it wants to give up its host role, it may firstly set PSP bit in USBFS_HPCS register to make the USB bus enter in suspend status. Then, the B-Device will enter in suspend state 3ms later. If the B-Device wants to change to be a host, HNPREQ bit in USBFS_GOTGCS register should be set and the USBFS will begin to perform HNP protocol on bus, and at last, the result of HNP is reported in HNPS bit in USBFS_GOTGCS register. Besides, it is always available to get the current role (host or device) from COPM bit in USBFS_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 should be reported in ASV and BSV bits in USBFS_GOTGCS register.

Set SRPREQ bit in USBFS_GOTGCS register to start a SRP request when USBFS is in B-Device OTG mode and USBFS will generate a success flag SRPS in USBFS_GOTGCS register if the SRP request successfully.

When USBFS is in OTG A-Device mode and it has detected a SRP request from a B-Device, it sets a SESIF flag in USBFS_GINTF register. The 5V power supply for VBUS pin should be prepared to switch on after getting this flag.

33.5.5. Data FIFO

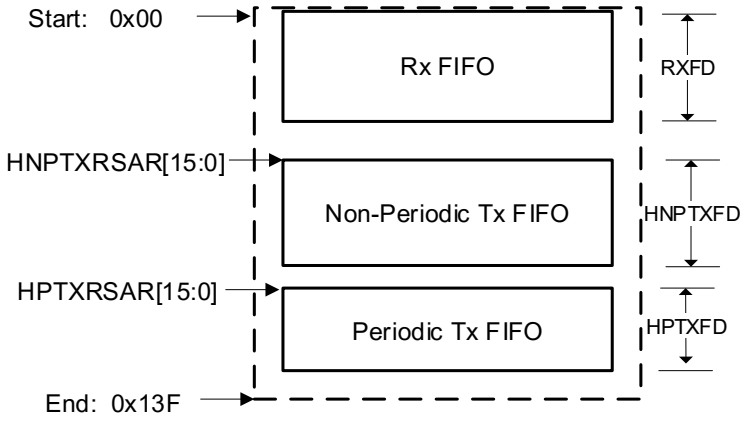
The USBFS contains a 1.25K bytes data FIFO for packet data storage. The data FIFO is implemented by using an internal SRAM in USBFS.

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 packets reception. All the periodic OUT channels share the periodic Tx FIFO to packets transmission. All the non-periodic OUT channels share the non-Periodic Tx FIFO for transmit packets. The size and start offset of these data FIFOs should be configured using these registers: USBFS_GRFLEN, USBFS_HNPTFLEN and USBFS_HPTFLEN. ***Figure 33-5. HOST mode FIFO space in SRAM*** describes the structure of these FIFOs in SRAM. The values in the figure are in term

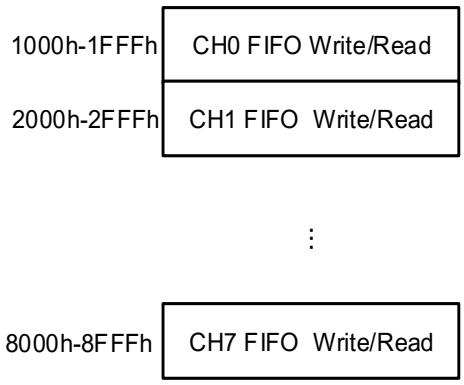
of 32-bit words.

Figure 33-5. HOST mode FIFO space in SRAM



USBFS provides a special register area for the internal data FIFO reading and writing. ***Figure 33-6. Host mode FIFO access register map*** describes the register memory area that the data FIFO can write. This area can be read by any channel data FIFO. The addresses in the figure are addressed in bytes. Each channel has its own FIFO access register space, although all Non-periodic channels share the same FIFO and all the Periodic channels also share the same FIFO. It is important for USBFS to know which channel the current pushed packet belongs to. Rx FIFO is also able to be accessed using USBFS_GRSTATR / USBFS_GRSTATP register.

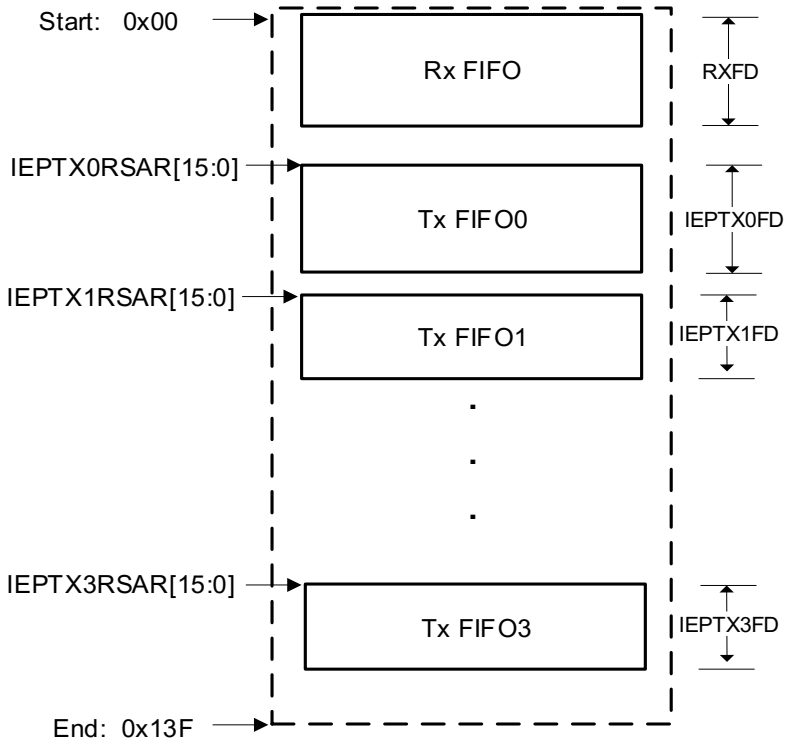
Figure 33-6. Host mode FIFO access register map



Device mode

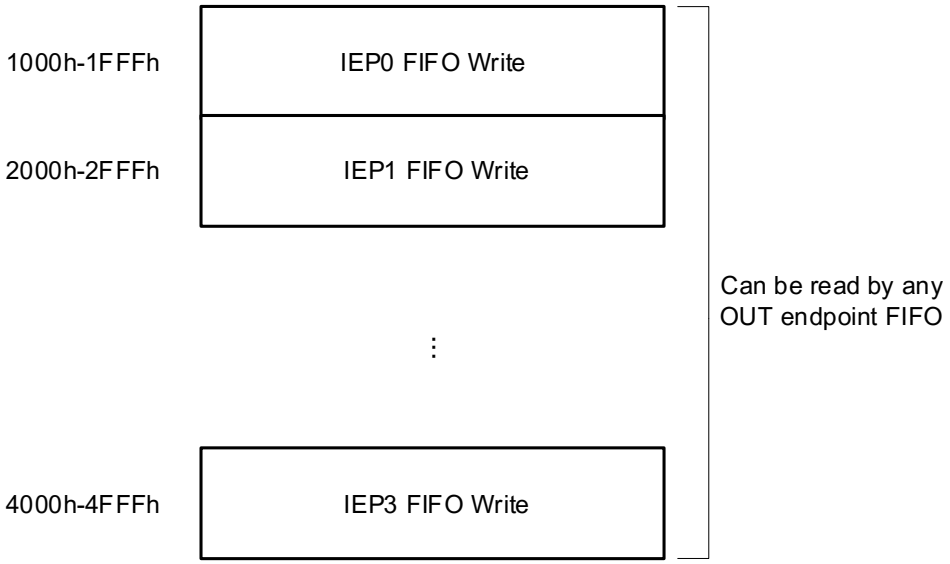
In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 4 Tx FIFOs (one for each IN endpoint). All the OUT endpoints share the Rx FIFO for receiving packets. The size and start offset of these data FIFOs should be configured using USBFS_GRFLEN and USBFS_DIEPxTFLEN (x=0…3) registers. ***Figure 33-7. 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 33-7. Device mode FIFO space in SRAM



USBFS provides a special register area for the internal data FIFO reading and writing. ***Figure 33-8. Device mode FIFO access register map*** describes the register memory area where the data FIFO can write. This area can be read by any endpoint FIFO. The addresses in the figure are addressed in bytes. Each endpoint has its own FIFO access register space. Rx FIFO is also able to be accessed using USBFS_GRSTATR/USBFS_GRSTATP register.

Figure 33-8. Device mode FIFO access register map



33.5.6. Operation guide

This section describes the advised operation guide for USBFS.

Host mode

Global register initialization sequence

1. Program USBFS_GAHBCS register according to application's demand, such as the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBFS_GUSBCS register according to application's demand, such as the operation mode (host, device or OTG) and some parameters of OTG and USB protocols.
3. Program USBFS_GCCFG register according to application's demand.
4. Program USBFS_GRFLEN, USBFS_HNPTFLEN_DIEP0TFLEN and USBFS_HPTFLEN register to configure the data FIFOs according to application's demand.
5. Program USBFS_GINTEN register to enable Mode Fault and Host Port interrupt and set GINTEN bit in USBFS_GAHBCS register to enable global interrupt.
6. Program USBFS_HPCS register and set PP bit.
7. Wait for a device's connection, and once a device is connected, the connection interrupt PCD in USBFS_HPCS register will be triggered. Then set PRST bit to perform a port reset. Wait for at least 10ms and then clear PRST bit.
8. Wait PEDC interrupt in USBFS_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 USBFS_HFT register to change the SOF interval if needed.

Channel initialization and enable sequence

1. Program USBFS_HCHxCTL registers with desired transfer type, direction, packet size, etc. Ensure that CEN and CDIS bits keep cleared during configuration.
2. Program USBFS_HCHxINTEN register. Set the desired interrupt enable bits.
3. Program USBFS_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 USBFS_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, TLEN can be set to a maximum possible value supported by Rx FIFO.

4. Set CEN bit in USBFS_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. USBFS will generate a channel disable request entry in request queue after the register setting operation. When the request entry reaches the top of request queue, it is processed by USBFS immediately:

For OUT channels, the specified channel will be disabled immediately. Then, a CH flag will be generated and the CEN and CDIS bits will be cleared by USBFS.

For IN channels, USBFS 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

1. Initialize USBFS global registers.
2. Initialize the channel.
3. Enable the channel.
4. After the IN channel is enabled by software, USBFS generates an Rx request entry in the corresponding request queue.
5. When the Rx request entry reaches the top of the request queue, USBFS begins to process this request entry. If bus time for the IN transaction indicated by the request entry is enough, USBFS starts the IN transaction on USB bus.
6. If the IN transaction finishes successfully (ACK handshake received), USBFS pushes the received data packet into the Rx FIFO and triggers ACK flag. Otherwise, the status flag (NAK) reports the transaction result.
7. If the IN transaction described in step 5 is successful and PCNT is larger than 1 in step2, return to step 3 and continues to receive the remaining packets. If the IN transaction described in step 5 is not successful, return to step 3 to re-receive the packet again.
8. After all the transactions in a transfer are successfully received on USB bus, USBFS pushes a TF status entry into the Rx FIFO on top of the last packet data. Thus after reading and popping all the received data packet, the TF status entry is need, USBFS 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.

OUT transfers operation sequence

1. Initialize USBFS 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, USBFS generates a Tx request entry in the corresponding request queue and decreases the TLEN field in USBFS_HCHxLEN register by the written packet's size.

4. When the request entry reaches the top of the request queue, USBFS begins to process this request entry. If bus time for the transaction indicated by the request entry is enough, USBFS starts the OUT transaction on USB bus.
5. When the OUT transaction indicated by the request entry finishes on USB bus, PCNT in USBFS_HCHxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flag (NAK) reports the transaction result.
6. If the OUT transaction described in step 5 is successful and PCNT is larger than 1 in step2, return to step 3 and continues to send the remaining packets. If the OUT transaction described in step 5 is not successful, return to step 3 to resend the packet again.
7. After all the transactions in a transfer are successfully sent on USB bus, USBFS 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.

Device mode

Global register initialization sequence

1. Program USBFS_GAHBCS register according to application's demand, such as the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBFS_GUSBCS register according to application's demand, such as: the operation mode (host, device or OTG) and some parameters of OTG and USB protocols.
3. Program USBFS_GCCFG register according to application's demand.
4. Program USBFS_GRFLEN, USBFS_HNPTFLEN_DIEP0TFLEN, USBFS_DIEPxTFLEN register to configure the data FIFOs according to application's demand.
5. Program USBFS_GINTEN register to enable Mode Fault, Suspend, SOF, Enumeration Done and USB Reset interrupt and then, set GINTEN bit in USBFS_GAHBCS register to enable global interrupt.
6. Program USBFS_DCFG register according to application's demand, such as the 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 USBFS_GINTF register.
8. Wait for ENUMF interrupt in USBFS_GINTF register.

Endpoint initialization and enable sequence

1. Program USBFS_DIEPxCTL or USBFS_DOEPxCTL register with desired transfer type, packet size, etc.
2. Program USBFS_DIEPINTEN or USBFS_DOEPINTEN register. Set the desired interrupt enable bits.
3. Program USBFS_DIEPxLEN or USBFS_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 USBFS_DIEPxCTL register, and the last packet's size is calculated based on PCNT, TLEN and MPL. If a zero-length packet is required to be sent, 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, TLEN can be set to a maximum possible value supported by Rx FIFO.
4. Set EPEN bit in USBFS_DIEPxCTL or USBFS_DOEPxCTL register to enable the endpoint.

Endpoint disable sequence

The endpoint can be disabled anytime when the EPEN bit in USBFS_DIEPxCTL or USBFS_DOEPxCTL registers is cleared.

IN transfers operation sequence

1. Initialize USBFS 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, USBFS decreases the TLEN field in USBFS_DIEPxLEN register by the written packet's size.
4. When an IN token received, USBFS transmits the data packet, and after the transaction finishes on USB bus, PCNT in USBFS_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags reports the transaction result.
5. After all the data packets in a transfer are successfully sent on USB bus, USBFS generates TF flag to indicate that the transfer successfully finishes and disables the IN endpoint.

OUT transfers operation sequence

1. Initialize USBFS global registers.
2. Initialize the endpoint and enable the endpoint.

3. When an OUT token received, USBFS receives the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBFS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBFS_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 successfully received on USB bus, USBFS pushes a TF status entry into the Rx FIFO on top of the last packet data. Thus after reading and popping all the received data packet, the TF status entry is read, USBFS generates TF flag to indicate that the transfer successfully finishes and disables the OUT endpoint.

33.6. Interrupts

USBFS has two interrupts: global interrupt and wake-up interrupt.

The source flags of the global interrupt are readable in USBFS_GINTF register and are listed in *Table 33-2. USBFS global interrupt*.

Table 33-2. USBFS global interrupt

Interrupt Flag	Description	Operation Mode
SESIF	Session interrupt	Host or device mode
DISCIF	Disconnect interrupt flag	Host Mode
IDPSC	ID pin status change	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
PXNCIF / ISOONCIF	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

Interrupt Flag	Description	Operation 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 can be triggered when USBFS is in suspend state, even when the USBFS's clocks are stopped. The source of the wake-up interrupt is WKUPIF bit in USBFS_GINTF register.

33.7. Register definition

USBFS base address: 0x5000 0000

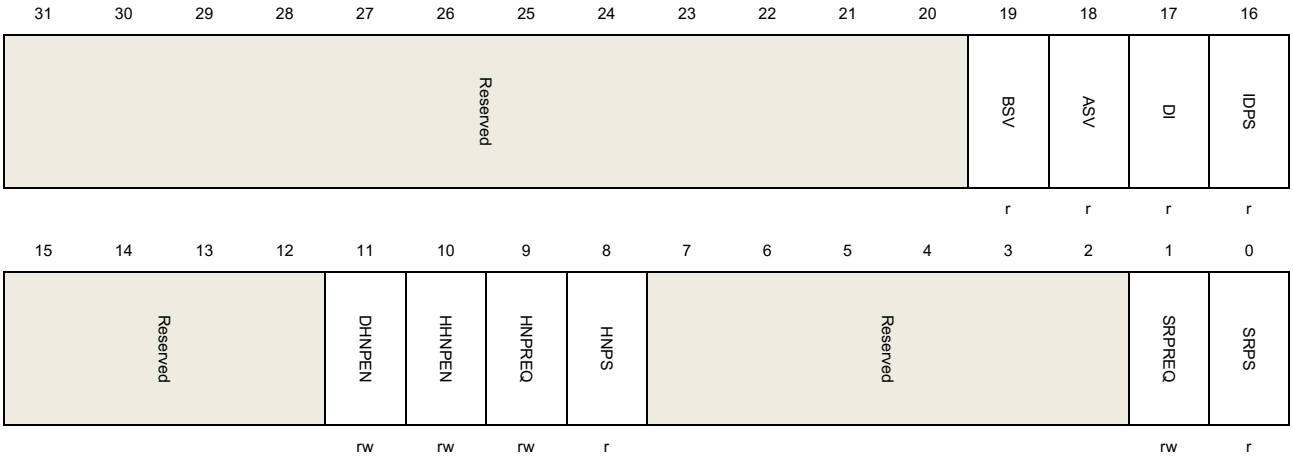
33.7.1. Global control and status registers

Global OTG control and status register (USBFS_GOTGCS)

Address offset: 0x0000

Reset value: 0x0000 0800

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



Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19	BSV	B-Session Valid (described in OTG protocol). 0: Vbus voltage level of a OTG B-Device is below VBSESSVLD 1: Vbus voltage level of a OTG B-Device is above VBSESSVLD **Note:** Only accessible in OTG B-Device mode.
18	ASV	A- Session valid A-host mode transceiver status. 0: Vbus voltage level of a OTG A-Device is below VASESSVLD 1: Vbus voltage level of a OTG A-Device is above VASESSVLD The A-Device is the default host at the start of a session. **Note:** Only accessible in OTG A-Device mode.
17	DI	Debounce interval Debounce interval of a detected connection. 0: Indicates the long debounce interval, when a plug-on and connection occurs on USB bus 1: Indicates the short debounce interval, when a soft connection is used in HNP protocol.

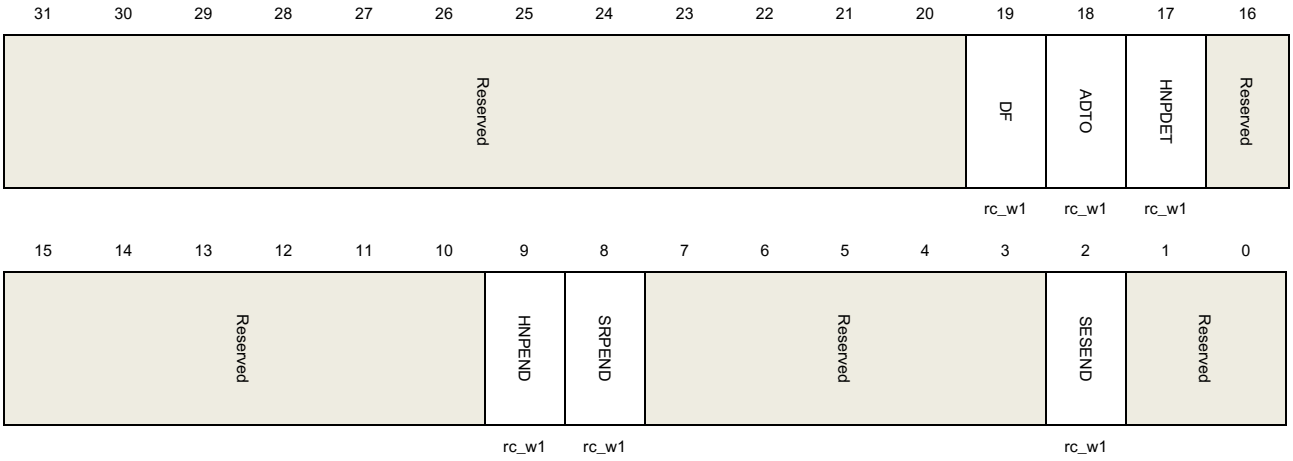
		Note: Only accessible in host mode.
16	IDPS	ID pin status Voltage level of connector ID pin 0: USBFS is in A-Device mode 1: USBFS is in B-Device mode **Note:** Accessible in both device and host modes.
15:12	Reserved	Must be kept at reset value.
11	DHNPEN	Device HNP enable Enable the HNP function of a B-Device. If this bit is cleared, USBFS doesn't start HNP protocol when application set HNPREQ bit in USBFS_GOTGCS register. 0: HNP function is not enabled. 1: HNP function is enabled **Note:** Only accessible in device mode.
10	HHNPEN	Host HNP enable Enable the HNP function of an A-Device. If this bit is cleared, USBFS doesn't response to the HNP request from B-Device. 0: HNP function is not enabled. 1: HNP function is enabled **Note:** Only accessible in host mode.
9	HNPREQ	HNP request This bit is set by software to start a HNP on the USB. This bit can be cleared when HNPEND bit in USBFS_GOTGINTF register is set, by writing zero to it, or clearing the HNPEND bit in USBFS_GOTGINTF register. 0: Don't send HNP request 1: Send HNP request **Note:** Only accessible in device mode.
8	HNPS	HNP successes This bit is set by the core when HNP succeeds, and this bit is cleared when HNPREQ bit is set. 0: HNP fails 1: HNP succeeds **Note:** Only accessible in device mode.
7:2	Reserved	Must be kept at reset value.
1	SRPREQ	SRP request This bit is set by software to start a SRP on the USB. This bit can be cleared when SRPEND bit in USBFS_GOTGINTF register is set, by writing zero to it, or clearing the SRPEND bit in USBFS_GOTGINTF register. 0: No session request 1: Session request **Note:** Only accessible in device mode.

0	SRPS	SRP success This bit is set by the core when SRP succeeds, and this bit is cleared when SRPREQ bit is set. 0: SRP fails 1: SRP succeeds **Note:** Only accessible in device mode.

Global OTG interrupt flag register (USBFS_GOTGINTF)

Address offset: 0x0004
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	DF	Denounce finish Set by USBFS when the denounce during device connection is done. **Note:** Only accessible in host mode.
18	ADTO	A-Device timeout Set by USBFS when the A-Device's waiting for a B-Device' connection has timed out. **Note:** Accessible in both device and host modes.
17	HNPDET	Host negotiation request detected Set by USBFS when A-Device detects a HNP request. **Note:** Accessible in both device and host modes.
16:10	Reserved	Must be kept at reset value.
9	HNPEND	HNP end Set by the core when a HNP ends. Read the HNPS in USBFS_GOTGCS register to get the result of HNP.

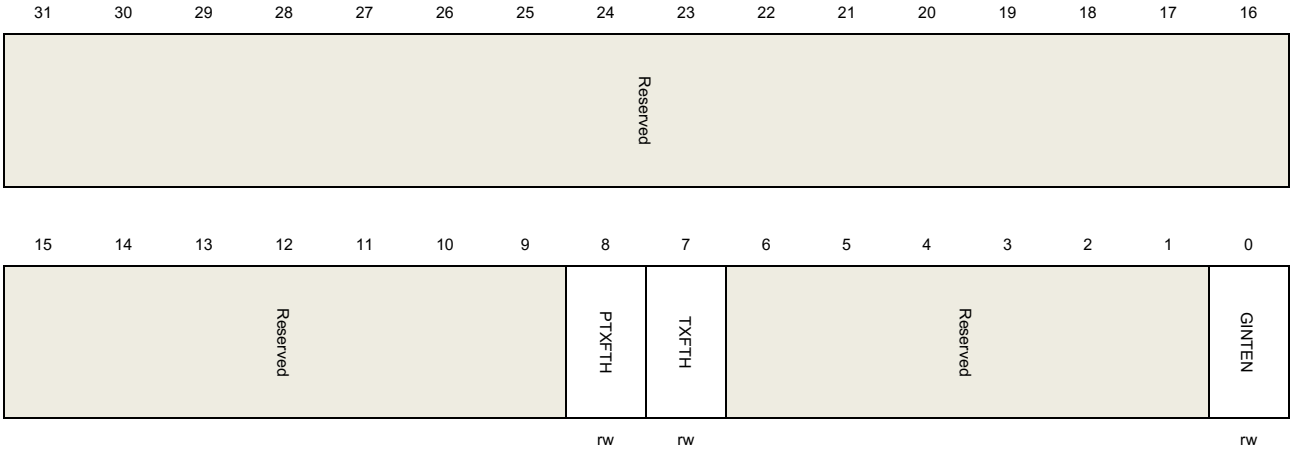
		Note: Accessible in both device and host modes.
8	SRPEND	SRPEND Set by the core when a SRP ends. Read the SRPS in USBFS_GOTGCS register to get the result of SRP. **Note:** Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value.
2	SESEND	Session end Set by the core when VBUS voltage is below Vb_ses_vld.
1:0	Reserved	Must be kept at reset value.

Global AHB control and status register (USBFS_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 0: PTXFEIF will be triggered when the periodic transmit FIFO is half empty 1: PTXFEIF will be triggered when the periodic transmit FIFO is completely empty **Note:** Only accessible in host mode.
7	TXFTH	Tx FIFO threshold Device mode: 0: TXFEIF will be triggered when the IN endpoint transmit FIFO is half empty 1: TXFEIF will be triggered when the IN endpoint transmit FIFO is completely empty Host mode: 0: NPTXFEIF will be triggered when the non-periodic transmit FIFO is half empty 1: NPTXFEIF will be triggered when the non-periodic transmit FIFO is completely

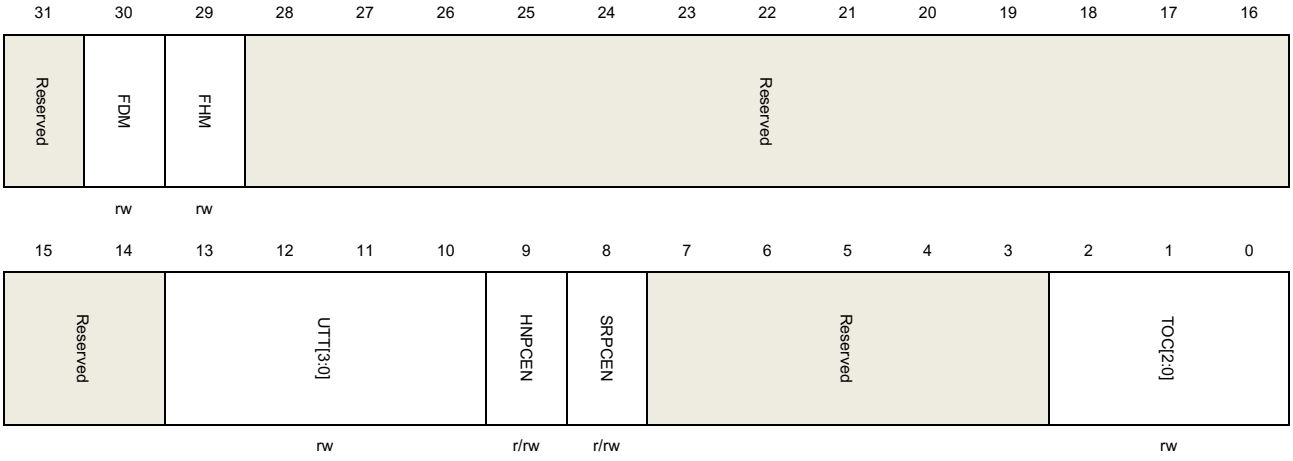
		empty
6:1	Reserved	Must be kept at reset value.
0	GINTEN	Global interrupt enable 0: Global interrupt is not enabled. 1: Global interrupt is enabled. **Note:** Accessible in both device and host modes.

Global USB control and status register (USBFS_GUSBCS)

Address offset: 0x000C

Reset value: 0x0000 0A80

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 Setting this bit will force the core to device mode irrespective of the USBFS ID input pin. 0: Normal mode 1: Device mode The application must wait at least 25 ms for the change taking effect after setting the force bit. **Note:** Accessible in both device and host modes.
29	FHM	Force host mode Setting this bit will force the core to host mode irrespective of the USBFS ID input pin. 0: Normal mode 1: Host mode The application must wait at least 25 ms for the change taking effect after setting

		the force bit. **Note:** Accessible in both device and host modes.
28:14	Reserved	Must be kept at reset value.
13:10	UTT[3:0]	USB turnaround time Turnaround time in PHY clocks. **Note:** Only accessible in device mode.
9	HNPCEN	HNP capability enable Controls whether the HNP capability is enabled 0: HNP capability is disabled 1: HNP capability is enabled **Note:** Accessible in both device and host modes.
8	SRPCEN	SRP capability enable Controls whether the SRP capability is enabled 0: SRP capability is disabled 1: SRP capability is enabled **Note:** Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value.
2:0	TOC[2:0]	Timeout calibration USBFS 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 48MHZ.).

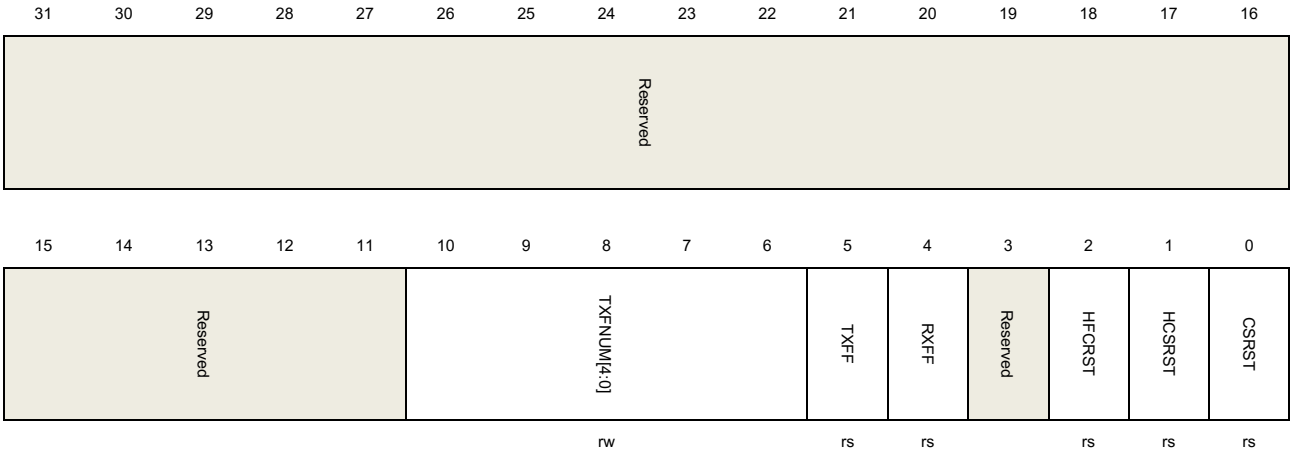
Global reset control register (USBFS_GRSTCTL)

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
31:11	Reserved	Must be kept at reset value.
10:6	TXFNUM[4:0]	Tx FIFO number Indicates which Tx FIFO will be flushed when TXFF bit in the same register is set. Host Mode: 00000: Only non-periodic Tx FIFO is flushed 00001: Only periodic Tx FIFO is flushed 1XXXX: Both periodic and non-periodic Tx FIFOs are flushed Other: Non data FIFO is flushed Device Mode: 00000: Only Tx FIFO0 is flushed 00001: Only Tx FIFO1 is flushed … 00011: Only Tx FIFO3 is flushed 1XXXX: All Tx FIFOs are flushed Other: Non data FIFO is flushed
5	TXFF	Tx FIFO flush Application set 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 USBFS. **Note:** Accessible in both device and host modes.
4	RXFF	Rx FIFO flush Application set 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 USBFS. **Note:** Accessible in both device and host modes.
3	Reserved	Must be kept at reset value.
2	HFCRST	Host frame counter reset Set by the application to reset the frame number counter in USBFS. 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 USBFS. **Note:** Only accessible in host mode.
1	HCSRST	HCLK soft reset Set by the application to reset AHB clock domain circuit. 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 USBFS. **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 (USBFS_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	Reserved.	PTXFEIF	HCIF	HPIF	Reserved		PXNCIF/ ISOONCIF	ISOINCIF	OEPIF	IEPIF	Reserved	
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 This interrupt is triggered when a resume signal (in device mode) or a remote wakeup signal (in host mode) is detected on the USB. **Note:** Accessible in both device and host modes.
30	SESIF	Session interrupt flag 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). **Note:** Accessible in both device and host modes.
29	DISCIF	Disconnect interrupt flag This interrupt is triggered after a device disconnection. Note: Only accessible in host mode.
28	IDPSC	ID pin status change Set by the core when ID status changes. **Note:** Accessible in both device and host modes.
27	Reserved	Must be kept at reset value.
26	PTXFEIF	Periodic Tx FIFO empty interrupt flag 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 USBFS_GAHBCS register. **Note:** Only accessible in host mode.

25	HCIF	Host channels interrupt flag Set by USBFS when one of the channels in host mode has raised an interrupt. First read USBFS_ HACHINT register to get the channel number, and then read the corresponding USBFS_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. **Note:** Only accessible in host mode.
24	HPIF	Host port interrupt flag Set by the core when USBFS detects that port status changes in host mode. Software should read USBFS_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. **Note:** Only accessible in host mode.
23:22	Reserved	Must be kept at reset value.
21	PXNCIF	Periodic transfer Not Complete Interrupt flag USBFS 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 At the end of a periodic frame (defined by EOPFT bit in USBFS_DCFG), USBFS will set this bit if there are still isochronous OUT endpoints for that not completed transactions. (Device Mode)
20	ISOINCIF	Isochronous IN transfer Not Complete Interrupt Flag At the end of a periodic frame (defined by EOPFT [1:0] bits in USBFS_DCFG), USBFS will set this bit if there are still isochronous IN endpoints for that not completed transactions. (Device Mode) **Note:** Only accessible in device mode.
19	OEPIF	OUT endpoint interrupt flag Set by USBFS when one of the OUT endpoints in device mode has raised an interrupt. Software should first read USBFS_DAEPINT register to get the device number, and then read the corresponding USBFS_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 Set by USBFS when one of the IN endpoints in device mode has raised an interrupt. Software should first read USBFS_DAEPINT register to get the device number, and then read the corresponding USBFS_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. Note: Only accessible in device mode.

17:16	Reserved	Must be kept at reset value.
15	EOPFIF	End of periodic frame interrupt flag When USB bus time in a frame reaches the value defined by EOPFT [1:0] bits in USBFS_DCFG register, USBFS sets this flag. **Note:** Only accessible in device mode.
14	ISOOPDIF	Isochronous OUT packet dropped interrupt flag USBFS 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. **Note:** Only accessible in device mode.
13	ENUMF	Enumeration finished USBFS sets this bit after the speed enumeration finishes. Read USBFS_DSTAT register to get the current device speed. **Note:** Only accessible in device mode.
12	RST	USB reset USBFS sets this bit when it detects a USB reset signal on bus. **Note:** Only accessible in device mode.
11	SP	USB suspend USBFS sets this bit when it detects that the USB bus is idle for 3 ms and enters suspend state. **Note:** Only accessible in device mode.
10	ESP	Early suspend USBFS sets this bit when it detects that the USB bus is idle for 3 ms. **Note:** Only accessible in device mode.
9:8	Reserved	Must be kept at reset value.
7	GONAK	Global OUT NAK effective Write 1 to SGONAK bit in the USBFS_DCTL register and USBFS will set GONAK flag after the writing to SGONAK takes effect. **Note:** Only accessible in device mode.
6	GNPINAK	Global Non-Periodic IN NAK effective Write 1 to SGINAK bit in the USBFS_DCTL register and USBFS will set GNPINAK flag after the writing to SGINAK takes effect. **Note:** Only accessible in device mode.
5	NPTXFEIF	Non-Periodic Tx FIFO empty interrupt flag 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 USBFS_GAHBCS register. **Note:** Only accessible in host mode.
4	RXFNEIF	Rx FIFO non-empty interrupt flag USBFS sets this bit when there is at least one packet or status entry in the Rx FIFO.

Note: Accessible in both host and device modes.

3	SOF	Start of frame Host Mode: USBFS sets this bit when it prepares to transmit a SOF or Keep-Alive on USB bus. Software can clear this bit by writing 1. Device Mode: USBFS sets this bit 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. **Note:** Accessible in both host and device modes.
2	OTGIF	OTG interrupt flag USBFS sets this bit when the flags in USBFS_GOTGINTF register generate an interrupt. Software should read USBFS_GOTGINTF register to get the source of this interrupt. This bit is cleared after the flags in USBFS_GOTGINTF causing this interrupt are cleared. **Note:** Accessible in both host and device modes.
1	MFIF	Mode fault interrupt flag USBFS sets this bit when software operates host-only register in device mode, or operates device-mode in host mode. These fault operations won't take effect. **Note:** Accessible in both host and device modes.
0	COPM	Current operation mode 0: Device mode 1: Host mode **Note:** Accessible in both host and device modes.

Global interrupt enable register (USBFS_GINTEN)

Address offset: 0x0018
Reset value: 0x0000 0000

This register works with the global interrupt flag register (USBFS_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	Reserved.	PTXFEIE	HCIE	HPIE	Reserved		PXNCIE/ ISOONCIE	ISOINCIE	OEPIE	IEPIE	Reserved	
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 0: Disable wakeup interrupt 1: Enable wakeup interrupt **Note:** Accessible in both host and device modes.
30	SESIE	Session interrupt enable 0: Disable session interrupt 1: Enable session interrupt **Note:** Accessible in both host and device modes.
29	DISCIE	Disconnect interrupt enable 0: Disable disconnect interrupt 1: Enable disconnect interrupt **Note:** Only accessible in device mode.
28	IDPSCIE	ID pin status change interrupt enable 0: Disable connector ID pin status interrupt 1: Enable connector ID pin status interrupt **Note:** Accessible in both host and device modes.
27	Reserved	Must be kept at reset value.
26	PTXFEIE	Periodic Tx FIFO empty interrupt enable 0: Disable periodic Tx FIFO empty interrupt 1: Enable periodic Tx FIFO empty interrupt **Note:** Only accessible in host mode.
25	HCIE	Host channels interrupt enable 0: Disable host channels interrupt 1: Enable host channels interrupt **Note:** Only accessible in host mode.
24	HPIE	Host port interrupt enable 0: Disable host port interrupt 1: Enable host port interrupt **Note:** Only accessible in host mode.
23:22	Reserved	Must be kept at reset value.
21	PXNCIE	Periodic transfer not complete Interrupt enable 0: Disable periodic transfer not complete interrupt 1: Enable periodic transfer not complete interrupt

		Note: Only accessible in host mode.
	ISOONCIE	Isochronous OUT transfer not complete interrupt enable 0: Disable isochronous OUT transfer not complete interrupt 1: Enable isochronous OUT transfer not complete interrupt **Note:** Only accessible in device mode.
20	ISOINCIE	Isochronous IN transfer not complete interrupt enable 0: Disable isochronous IN transfer not complete interrupt 1: Enable isochronous IN transfer not complete interrupt **Note:** Only accessible in device mode.
19	OEPIE	OUT endpoints interrupt enable 0: Disable OUT endpoints interrupt 1: Enable OUT endpoints interrupt **Note:** Only accessible in device mode.
18	IEPIE	IN endpoints interrupt enable 0: Disable IN endpoints interrupt 1: Enable IN endpoints interrupt **Note:** Only accessible in device mode.
17:16	Reserved	Must be kept at reset value.
15	EOPFIE	End of periodic frame interrupt enable 0: Disable end of periodic frame interrupt 1: Enable end of periodic frame interrupt **Note:** Only accessible in device mode.
14	ISOOPDIE	Isochronous OUT packet dropped interrupt enable 0: Disable isochronous OUT packet dropped interrupt 1: Enable isochronous OUT packet dropped interrupt **Note:** Only accessible in device mode.
13	ENUMFIE	Enumeration finish enable 0: Disable enumeration finish interrupt 1: Enable enumeration finish interrupt **Note:** Only accessible in device mode.
12	RSTIE	USB reset interrupt enable 0: Disable USB reset interrupt 1: Enable USB reset interrupt **Note:** Only accessible in device mode.
11	SPIE	USB suspend interrupt enable 0: Disable USB suspend interrupt 1: Enable USB suspend interrupt **Note:** Only accessible in device mode.
10	ESPIE	Early suspend interrupt enable

		0: Disable early suspend interrupt 1: Enable early suspend interrupt **Note:** Only accessible in device mode.
9:8	Reserved	Must be kept at reset value.
7	GONAKIE	Global OUT NAK effective interrupt enable 0: Disable global OUT NAK interrupt 1: Enable global OUT NAK interrupt **Note:** Only accessible in device mode.
6	GNPINAKIE	Global non-periodic IN NAK effective interrupt enable 0: Disable global non-periodic IN NAK effective interrupt 1: Enable global non-periodic IN NAK effective interrupt **Note:** Only accessible in device mode.
5	NPTXFEIE	Non-periodic Tx FIFO empty interrupt enable 0: Disable non-periodic Tx FIFO empty interrupt 1: Enable non-periodic Tx FIFO empty interrupt **Note:** Only accessible in Host mode.
4	RXFNEIE	Receive FIFO non-empty interrupt enable 0: Disable receive FIFO non-empty interrupt 1: Enable receive FIFO non-empty interrupt **Note:** Accessible in both device and host modes.
3	SOFIE	Start of frame interrupt enable 0: Disable start of frame interrupt 1: Enable start of frame interrupt **Note:** Accessible in both device and host modes.
2	OTGIE	OTG interrupt enable 0: Disable OTG interrupt 1: Enable OTG interrupt **Note:** Accessible in both device and host modes.
1	MFIE	Mode fault interrupt enable 0: Disable mode fault interrupt 1: Enable mode fault interrupt **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 (USBFS_GRSTATR/USBFS_GRSTATP)

Address offset for Read: 0x001C

Address offset for Pop: 0x0020

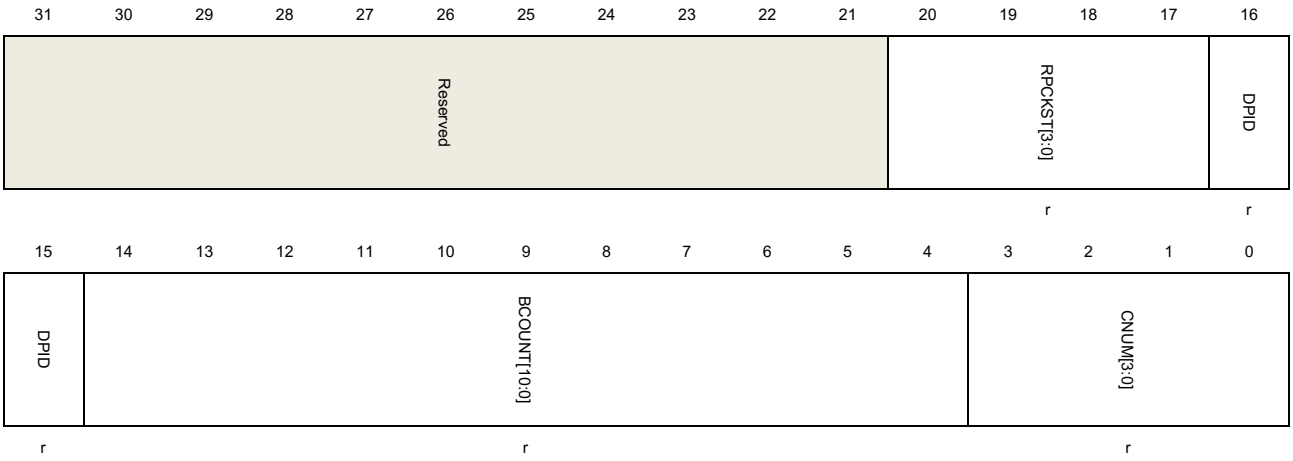
Reset value: 0x0000 0000

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 USBFS_GINTF) is triggered.

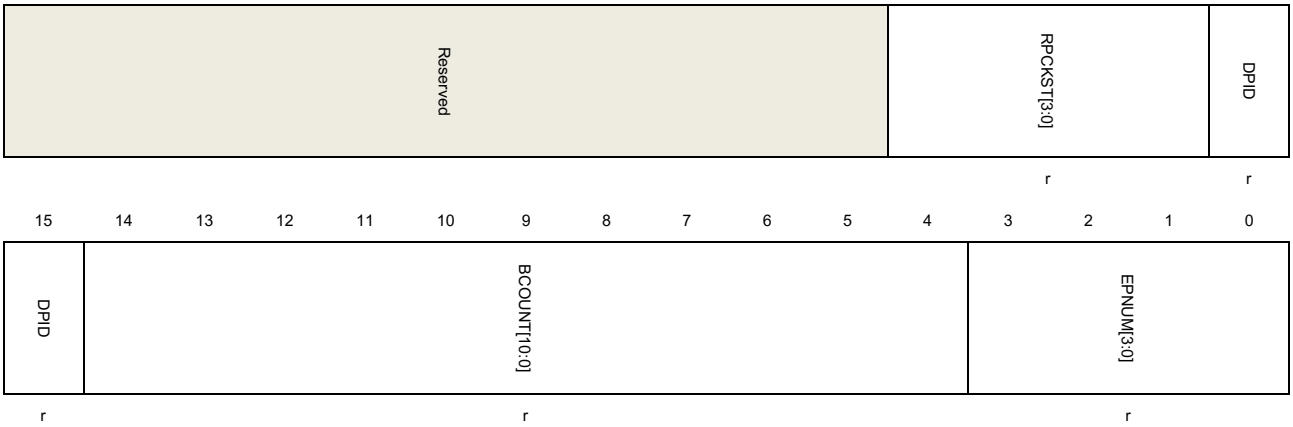
This register has to be accessed by word (32-bit)

Host mode:



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20:17	RPCKST[3:0]	Received packet status 0010: IN data packet received 0011: IN transfer completed (generates an interrupt if popped) 0101: Data toggle error (generates an interrupt if popped) 0111: Channel halted (generates an interrupt if popped) Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received packet 00: DATA0 10: DATA1 Others: Reserved
14:4	BCOUNT[10:0]	Byte count The byte count of the received IN data packet.
3:0	CNUM[3:0]	Channel number The channel number to which the current received packet belongs.

Device mode:



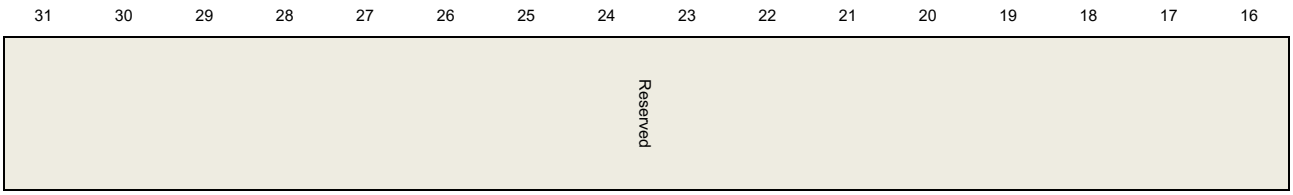
Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20:17	RPCKST[3:0]	Received packet status 0001: Global OUT NAK (generates an interrupt) 0010: OUT data packet received 0011: OUT transfer completed (generates an interrupt) 0100: SETUP transaction completed (generates an interrupt) 0110: SETUP data packet received Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received OUT data packet 00: DATA0 10: DATA1 Others: Reserved
14:4	BCOUNT[10:0]	Byte count 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.

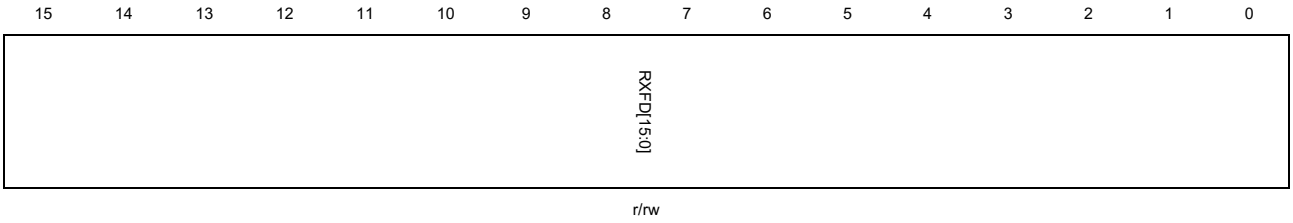
Global receive FIFO length register (USBFS_GRFLEN)

Address offset: 0x024

Reset value: 0x0000 0200

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



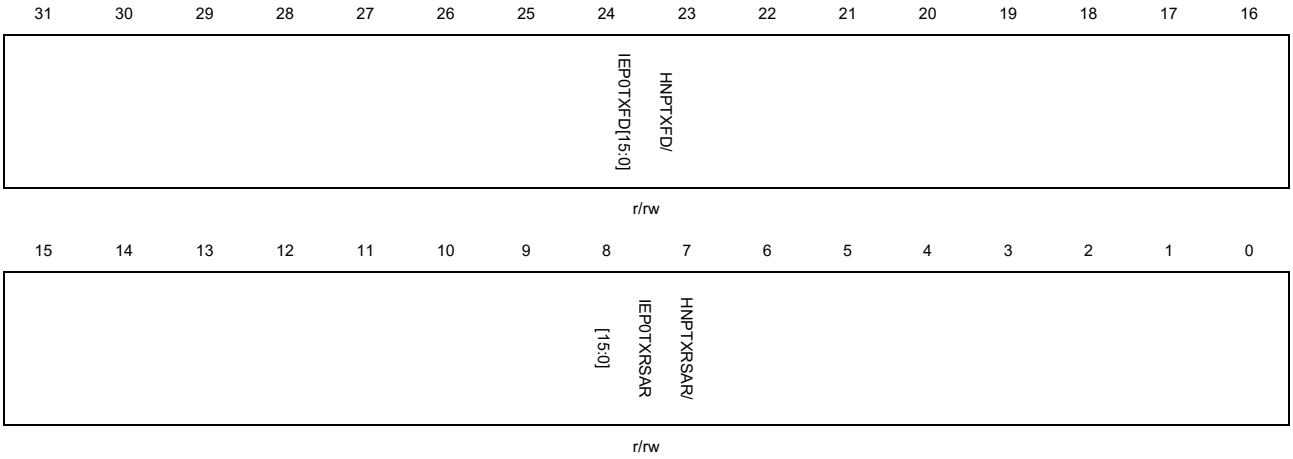


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RXFD[15:0]	Rx FIFO depth In terms of 32-bit words. 1<=RXFD<=1024

Host non-periodic transmit FIFO length register /Device IN endpoint 0 transmit FIFO length (USBFS_HNPTFLEN _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]	Host Non-periodic Tx FIFO depth In terms of 32-bit words. 1<=HNPTXFD<=1024
15:0	HNPTXRSAR[15:0]	Host Non-periodic Tx RAM start address The start address for non-periodic transmit FIFO RAM is in term of 32-bit words.

Device Mode:

Bits	Fields	Descriptions
31:16	IEP0TXFD[15:0]	IN Endpoint 0 Tx FIFO depth

		In terms of 32-bit words. 16<=IEP0TXFD<=140
15:0	IEP0TXRSAR[15:0]	IN Endpoint 0 TX RAM start address The start address for endpoint0 transmit FIFO RAM is in term of 32-bit words.

Host non-periodic transmit FIFO/queue status register (USBFS_HNPTFQSTAT)

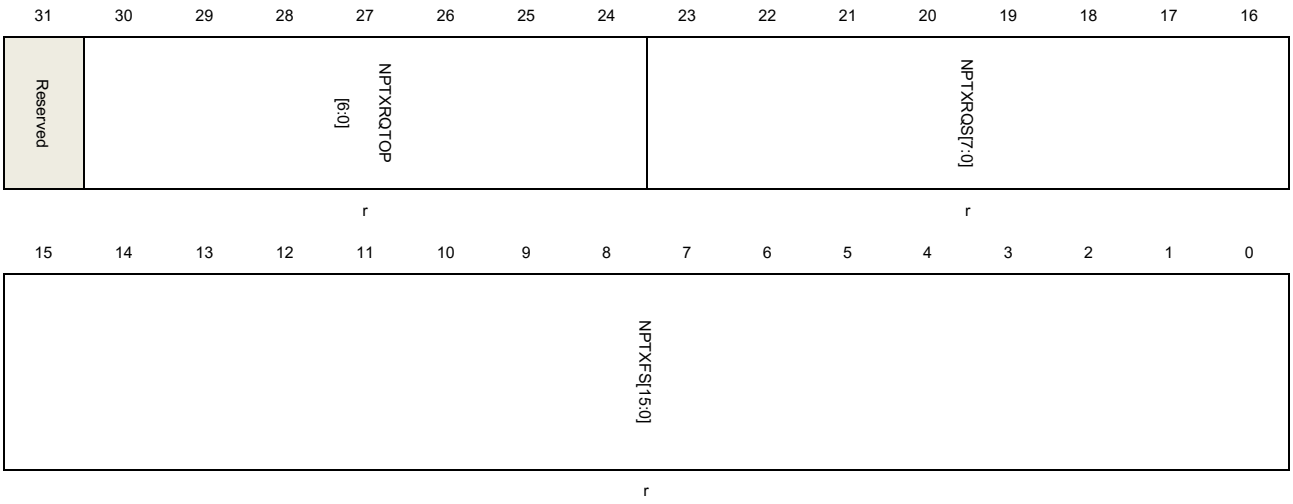
Address offset: 0x002C

Reset value: 0x0008 0200

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.

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



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 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 The remaining space of the non-periodic transmit request queue. 0: Request queue is Full 1: 1 entry

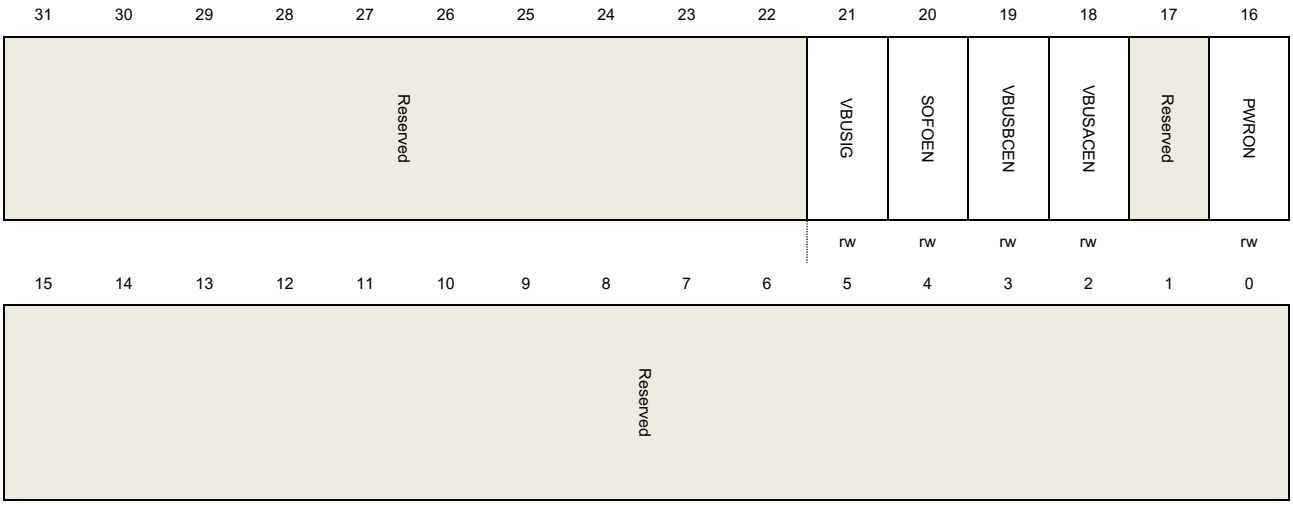
		2: 2 entries … n: n entries (0<=n<=8) Others: Reserved
15:0	NPTXFS[15:0]	Non-periodic Tx FIFO space The remaining space of the non-periodic transmit FIFO. In terms of 32-bit words. 0: Non-periodic Tx FIFO is full 1: 1 word 2: 2 words n: n words (0<=n<=NPTXFD) Others: Reserved

Global core configuration register (USBFS_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	VBUSIG	VBUS ignored When this bit is set, USBFS doesn't monitor the voltage on VBUS pin and always consider VBUS voltage as valid both in host mode and in device mode, then free the VBUS pin for other usage. 0: VBUS is not ignored. 1: VBUS is ignored and always consider VBUS voltage as valid.
20	SOFOEN	SOF output enable 0: SOF pulse output disabled.

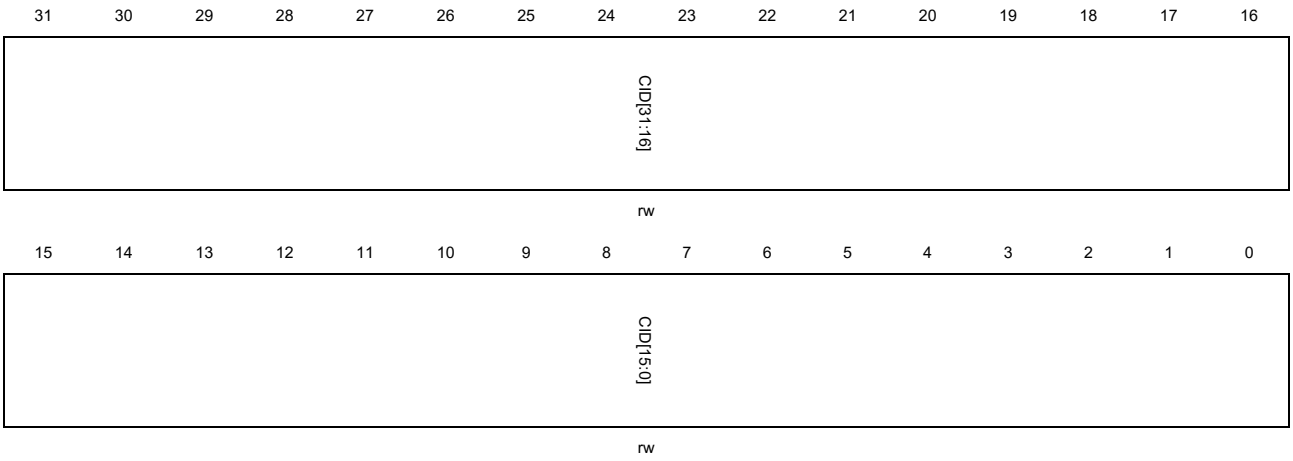
		1: SOF pulse output enabled.
19	VBUSBCEN	The V_{BUS} B-device Comparer enable 0: V_{BUS} B-device comparer disabled 1: V_{BUS} B-device comparer enabled
18	VBUSACEN	The VBUS A-device Comparer enable 0: V_{BUS} A-device comparer disabled 1: V_{BUS} A-device comparer enabled
17	Reserved	Must be kept at reset value.
16	PWRON	Power on This bit is the power switch for the internal embedded Full-Speed PHY. 0: Embedded Full-Speed PHY power off. 1: Embedded Full-Speed PHY power on.
15:0	Reserved	Must be kept at reset value.

Core ID register (USBFS_CID)

Address offset: 0x003C
Reset value: 0x0000 1000

This register contains the Product ID.

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



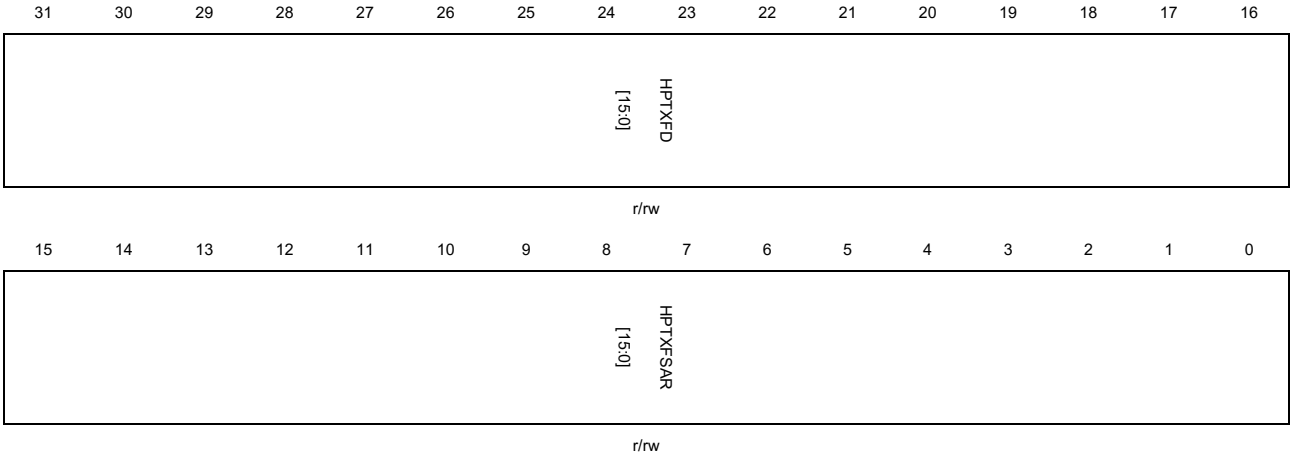
Bits	Fields	Descriptions
31:0	CID[31:0]	Core ID Software can write or read this field and uses this field as a unique ID for its application

Host periodic transmit FIFO length register (USBFS_HPTFLEN)

Address offset: 0x0100

Reset value: 0x0200 0600

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



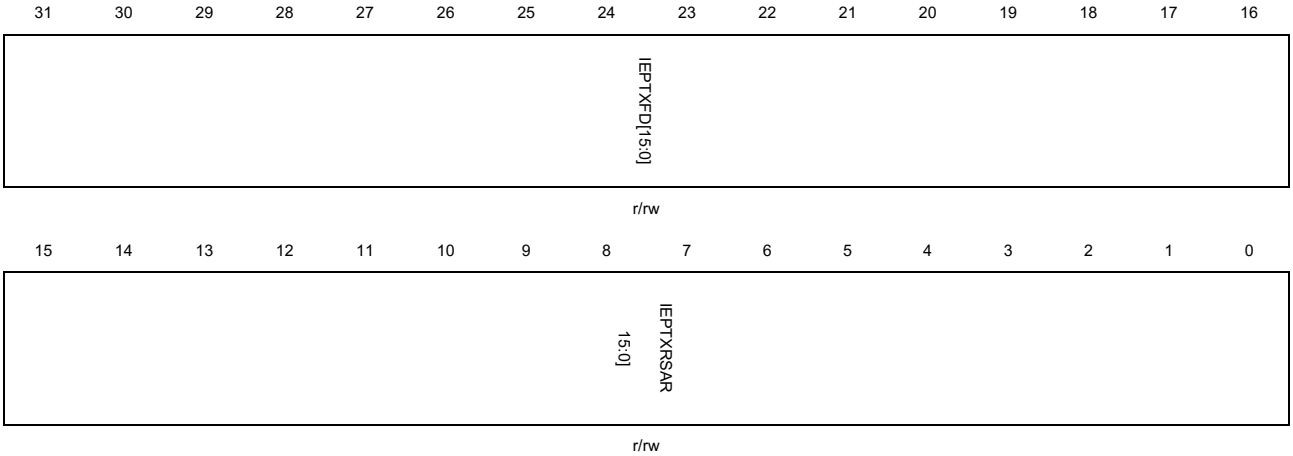
Bits	Fields	Descriptions
31:16	HPTXFD[15:0]	Host Periodic Tx FIFO depth In terms of 32-bit words. 1<=HPTXFD<=1024
15:0	HPTXFSAR[15:0]	Host periodic Tx FIFO RAM start address The start address for host periodic transmit FIFO RAM is in term of 32-bit words.

Device IN endpoint transmit FIFO length register (USBFS_DIEPxTFLEN) (x = 1 .. 3, 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 depth In terms of 32-bit words. 1<=HPTXFD<=1024
15:0	IEPTXRSAR[15:0]	IN endpoint FIFO Tx RAM start address The start address for IN endpoint transmit FIFOx is in term of 32-bit words.

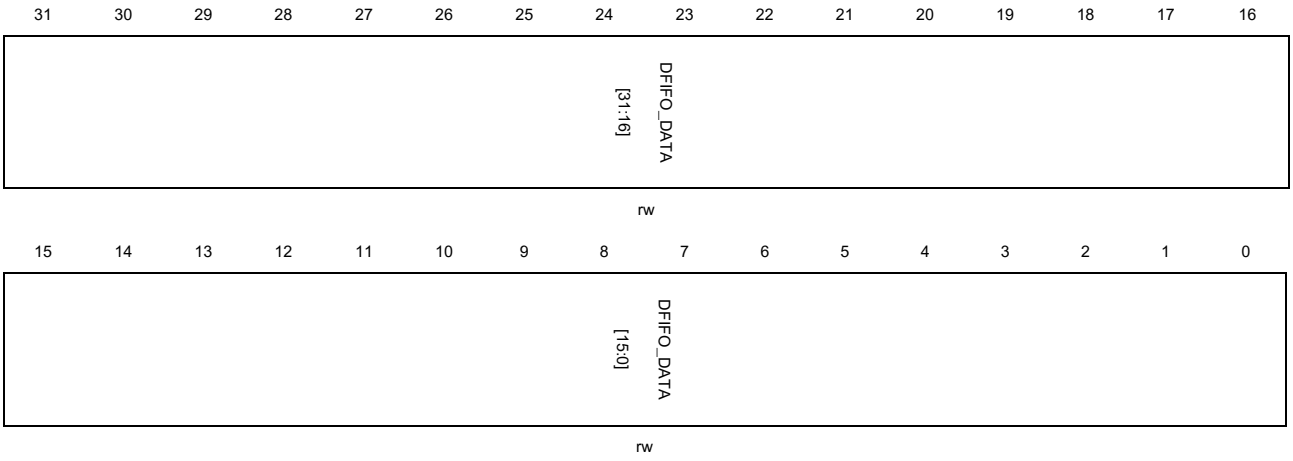
Data FIFO register (USBFS_DFIFO)

Address offset: 0x1000

Host mode: write address range: [0x1000,0x8FFF], read address range: [0x1000, 0xFFFF]

Device mode: write address range: [0x1000,0x4FFF], read address range: [0x1000, 0xFFFF]

Reset value: 0x0000 0000



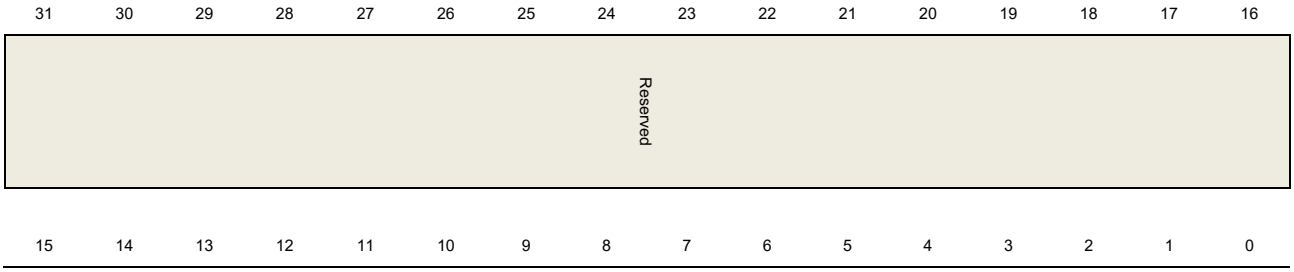
Bits	Fields	Descriptions
31:0	DFIFO_DATA[31:0]	Write the field would push the data into corresponding data FIFO，and read the field would pop the data out corresponding RX FIFO, if read the overflowing address, the last pop data would be gained.

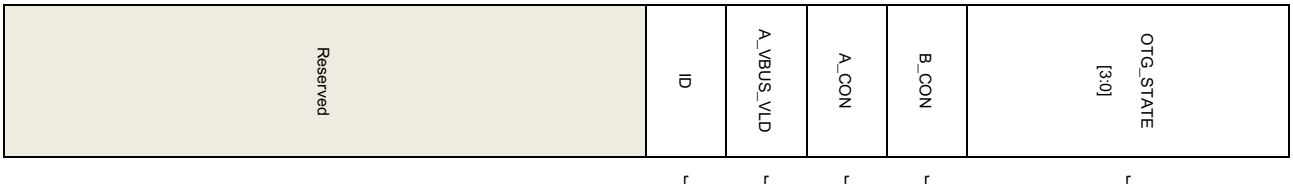
USBFS Debug register (USBFS_DBG)

Address offset: 0x10000

address range: [0x10000, 0x1FFFF]

Reset value: 0x0000 0000





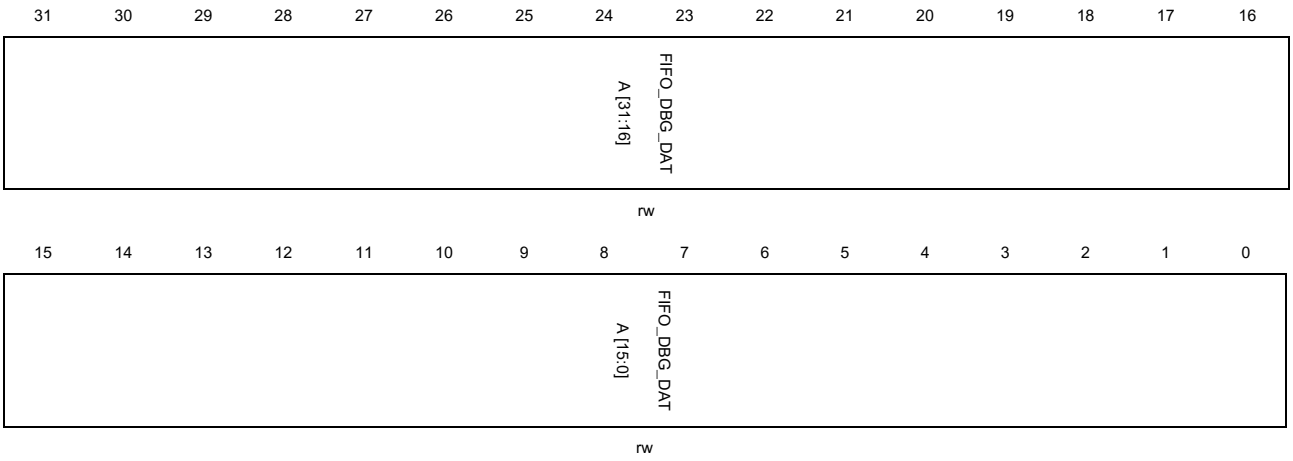
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	ID	ID signal
6	A_VBUS_VLD	Check whether the VBUS voltage exceed valid threshold value
5	A_CON	Detect A device is connected
4	B_CON	Detect B device is connected
3:0	OTG_STATE[3:0]	Internal OTG state of state machine

Data FIFO Debug register (USBFS_DFIFODBG)

Address offset: 0x20000

write address range: [0x20000, 0x204FF], read address range: [0x20000, 0x2FFFF]

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:0	FIFO_DBG_DATA[31:0]	The register is used to debug FIFO data, not Recommend to users. Write the field would push the data into corresponding data FIFO，and read the field would pop the data out corresponding address, if read the overflowing address range, the last pop data would be gained.

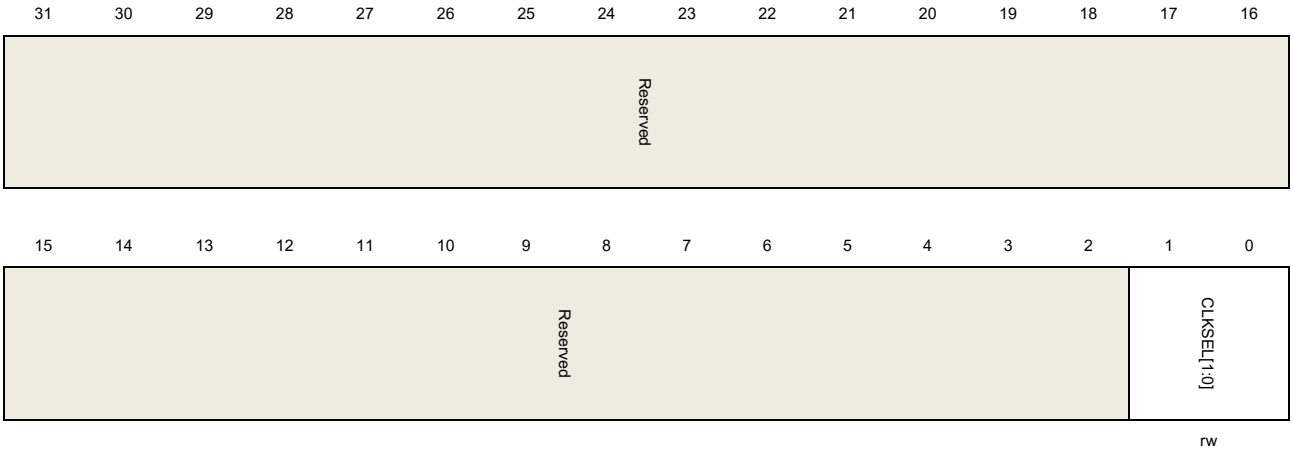
33.7.2. Host control and status registers

Host control register (USBFS_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)



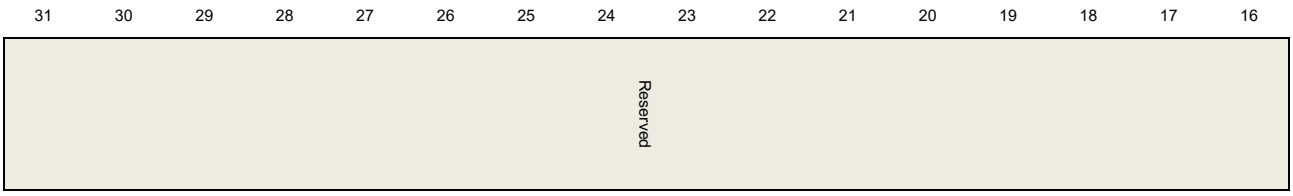
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1:0	CLKSEL[1:0]	Clock select for usb clock. 01: 48MHz clock others: reserved

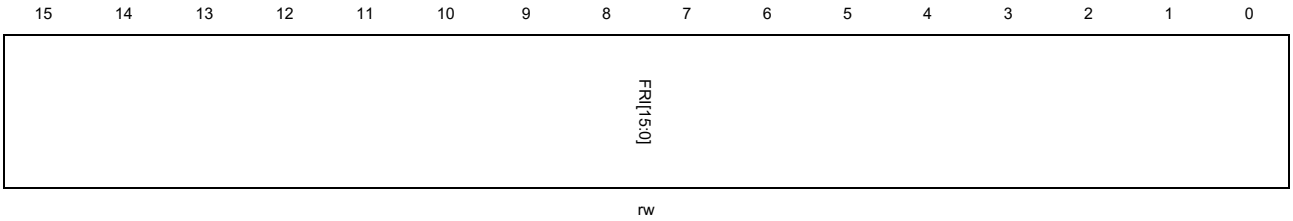
Host frame interval register (USBFS_HFT)

Address offset: 0x0404
Reset value: 0x0000 BB80

This register sets the frame interval for the current enumerating speed when USBFS controller is enumerating.

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





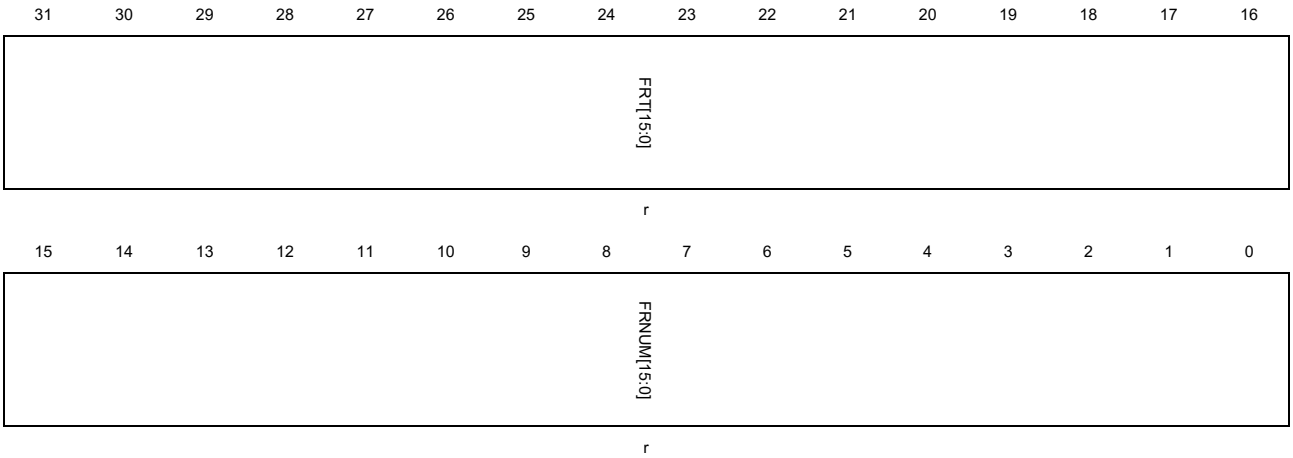
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	FRI[15:0]	Frame interval This value describes the frame time in terms of PHY clocks. Each time when port is enabled after a port reset operation, USBFS use a proper value according to the current speed, and software can write to this field to change the value. This value should be calculated using the frequency described below: Full-Speed: 48MHz Low-Speed: 6MHz

Host frame information remaining register (USBFS_HFINFR)

Address offset: 0x408

Reset value: 0xBB80 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 clocks.
15:0	FRNUM[15:0]	Frame number This field reports the frame number of current frame and returns to 0 after it reaches 0x3FFF.

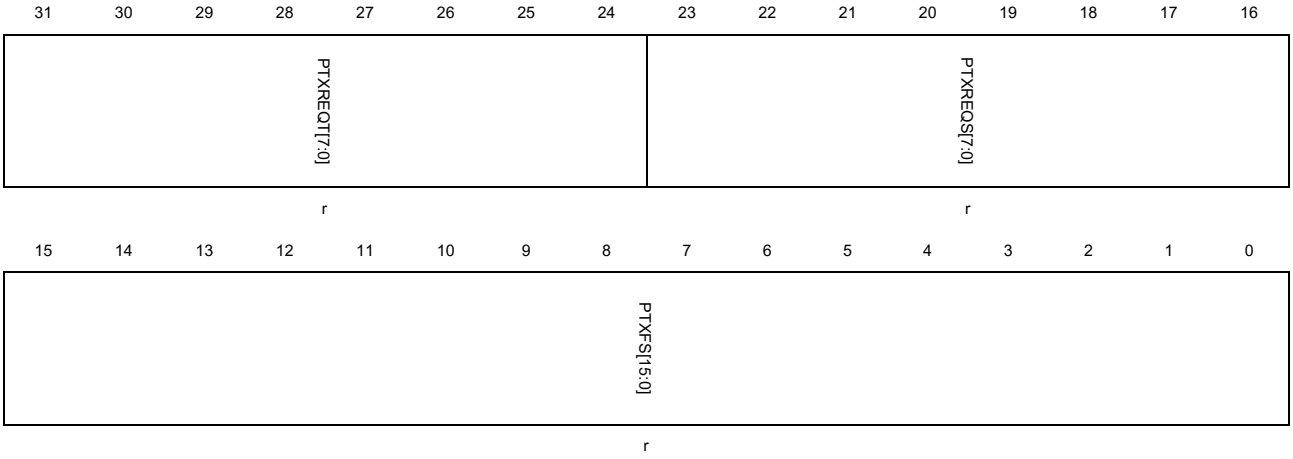
Host periodic transmit FIFO/queue status register (USBFS_HPTFQSTAT)

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 Entry in the 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	PTXREQS[7:0]	Periodic Tx request queue space The remaining space of the periodic transmit request queue. 0: Request queue is Full 1: 1 entry 2: 2 entries … n: n entries (0<=n<=8) Others: Reserved
15:0	PTXFS[15:0]	Periodic Tx FIFO space The remaining space of the periodic transmit FIFO. In terms of 32-bit words. 0: periodic Tx FIFO is full 1: 1 word

2: 2 words

n: n words (0<=n<=PTXFD)

Others: Reserved

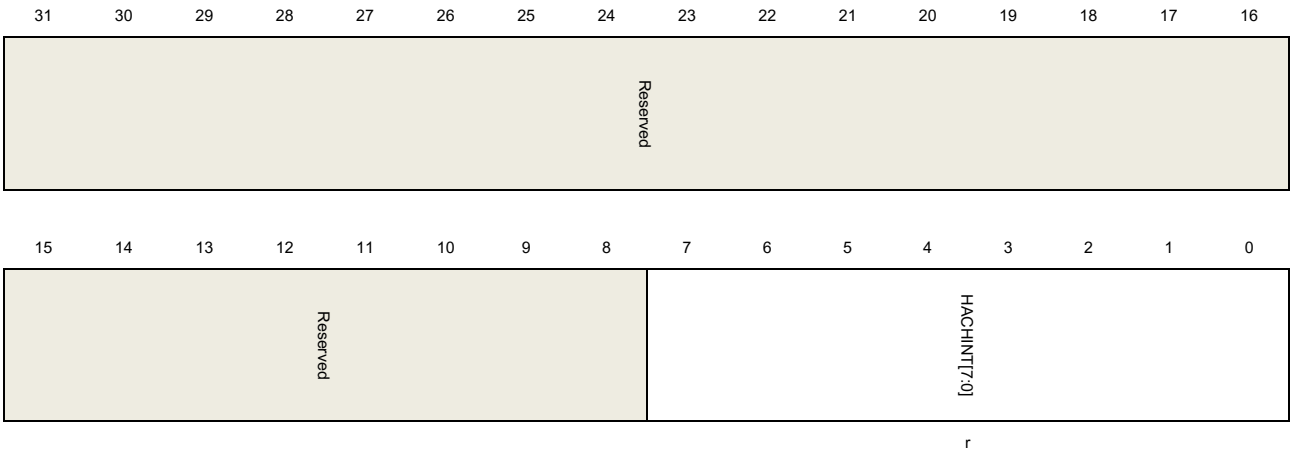
Host all channels interrupt register (USBFS_HACHINT)

Address offset: 0x0414

Reset value: 0x0000 0000

When a channel interrupt is triggered, USBFS set corresponding bit in this register and software should read this register to know which channel is asserting interrupts.

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



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	HACHINT[7:0]	Host all channel interrupts Each bit represents a channel: Bit 0 for channel 0, bit 7 for channel 7.

Host all channels interrupt enable register (USBFS_HACHINTEN)

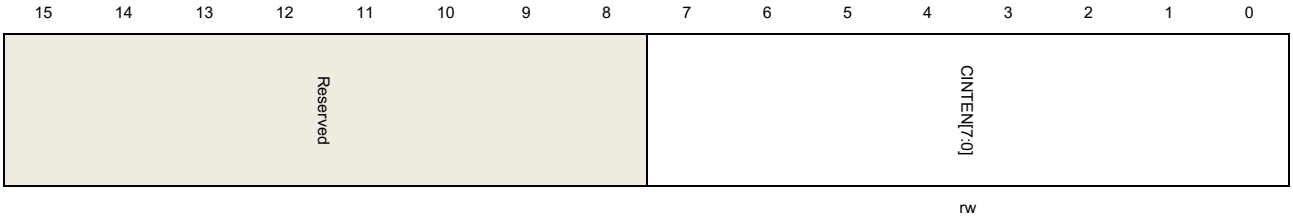
Address offset: 0x0418

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 USBFS_GINTF register.

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





Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	CINTEN[7:0]	Channel interrupt enable 0: Disable channel-n interrupt 1: Enable channel-n interrupt Each bit represents a channel: Bit 0 for channel 0, bit 7 for channel 7.

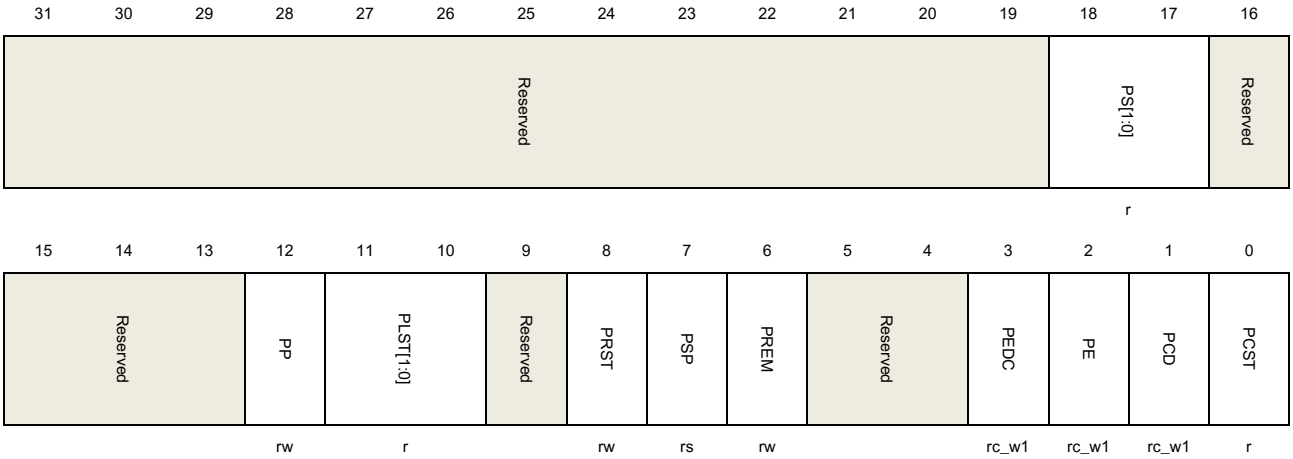
Host port control and status register (USBFS_HPCS)

Address offset: 0x0440

Reset value: 0x0000 0000

This register controls the port's behavior and also has some flags which report the status of the port. The HPIF flag in USBFS_GINTF register will be triggered if one of these flags in this register is set by USBFS: 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 Report the enumerated speed of the device attached to this port. 01: Full speed 10: Low speed

		Others: Reserved
16:13	Reserved	Must be kept at reset value.
12	PP	Port power This bit should be set before a port is used. Because USBFS 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 VBUS before setting this bit. 0: Port is powered off 1: Port is powered on
11:10	PLST[1:0]	Port line status Report the current state of USB data lines Bit 10: State of DP line Bit 11: State of DM line
9	Reserved	Must be kept at reset value.
8	PRST	Port reset 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. 0: Port is not in reset state 1: Port is in reset state
7	PSP	Port suspend 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: PRST bit in this register is set by application PREM bit in this register is set A remote wakeup signal is detected A device disconnect is detected 0: Port is not in suspend state 1: Port is in suspend state
6	PREM	Port resume 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. 0: No resume driven 1: Resume driven
5:4	Reserved	Must be kept at reset value.
3	PEDC	Port enable/disable change Set by the core when the status of the Port enable bit 2 in this register changes.
2	PE	Port Enable This bit is automatically set by USBFS after a USB reset signal finishes and cannot be set by software. This bit is cleared by the following events:

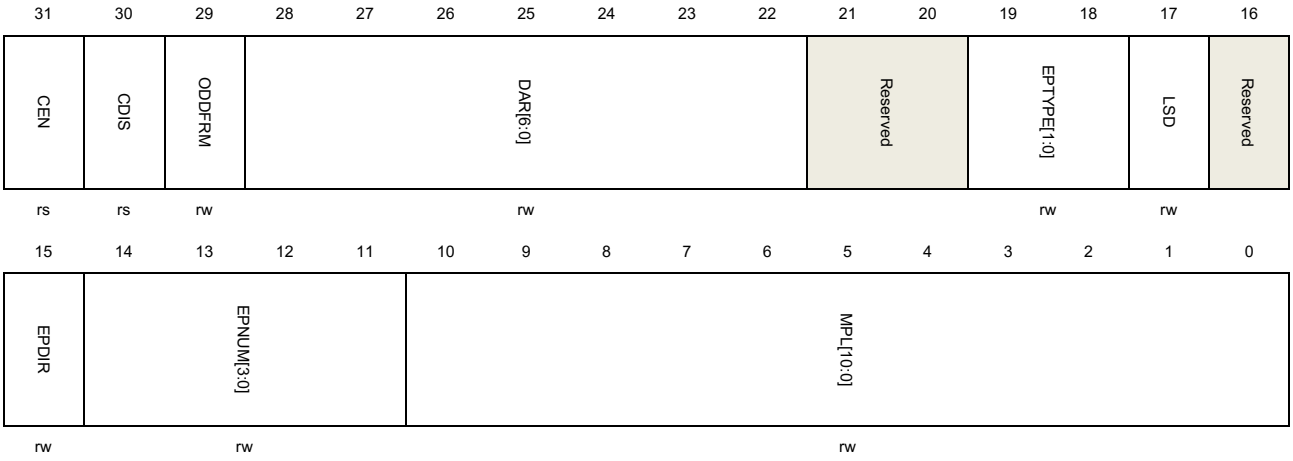
		A disconnect condition Software clearing this bit 0: Port disabled 1: Port enabled
1	PCD	Port connect detected Set by USBFS when a device connection is detected. This bit can be cleared by writing 1 to this bit.
0	PCST	Port connect status 0: Device is not connected to the port 1: Device is connected to the port

Host channel-x control register (USBFS_HCHxCTL) (x = 0 .. 7 where x = channel_number)

Address offset: 0x0500 + (channel_number × 0x20)

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	CEN	Channel enable Set by the application and cleared by USBFS. 0: Channel disabled 1: Channel enabled Software should following the operation guide to disable or enable a channel.
30	CDIS	Channel disable 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 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. 0: Even frame 1: Odd frame
28:22	DAR[6:0]	Device address The address of the USB device that this channel wants to communicate with.
21:20	Reserved	Must be kept at reset value.
19:18	EPTYPE[1:0]	Endpoint type The transfer type of the endpoint that this channel wants to communicate with. 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	LSD	Low-Speed device 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 The transfer direction of the endpoint that this channel wants to communicate with. 0: OUT 1: IN
14:11	EPNUM[3:0]	Endpoint number The number of the endpoint that this channel wants to communicate with.
10:0	MPL[10:0]	Maximum packet length The target endpoint's maximum packet length.

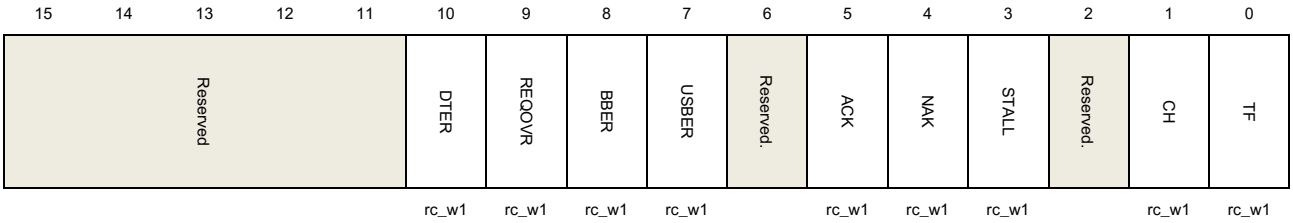
Host channel-x interrupt flag register (USBFS_HCHxINTF) (x = 0..7 where x = channel number)

Address offset: 0x0508 + (channel_number × 0x20)
Reset value: 0x0000 0000

This register contains the status and events of a channel, when software get 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 The IN transaction gets a data packet but the PID of this packet doesn't match DPID [1:0] bits in USBFS_HCHxLEN register.
9	REQOVR	Request queue overrun The periodic request queue is full when software starts new transfers.
8	BBER	Babble error 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 the endpoint's maximum packet length.
7	USBER	USB Bus Error The USB error flag is set when the following conditions occurs during receiving a packet: A received packet has a wrong CRC field A stuff error detected on USB bus Timeout when waiting for a response packet
6	Reserved	Must be kept at reset value.
5	ACK	ACK An ACK response is received or transmitted
4	NAK	NAK A NAK response is received.
3	STALL	STALL A STALL response is received.
2	Reserved	Must be kept at reset value.
1	CH	Channel halted This channel is disabled by a request, and it will not response to other requests during the request processing.
0	TF	Transfer finished All the transactions of this channel finish successfully, and no error occurs. For IN channel, this flag will be triggered after PCNT bits in USBFS_HCHxLEN register

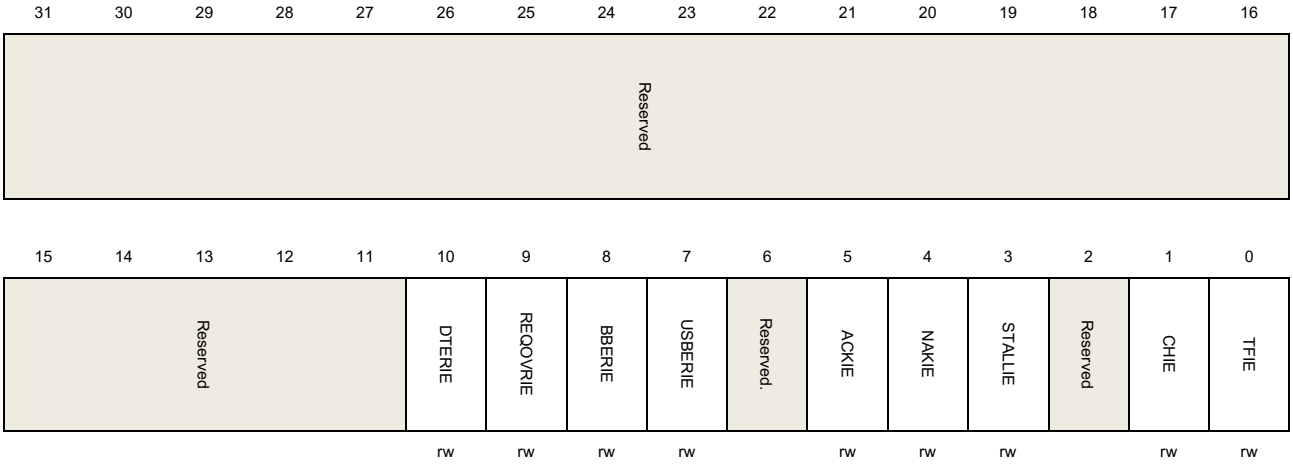
reach zero. For OUT channel, this flag will be triggered when software reads and pops a TF status entry from the RxFIFO.

Host channel-x interrupt enable register (USBFS_HCHxINTEN) (x = 0 .. 7, where x = channel number)

Address offset: 0x050C + (channel_number × 0x20)
Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS_HCHxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS_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)



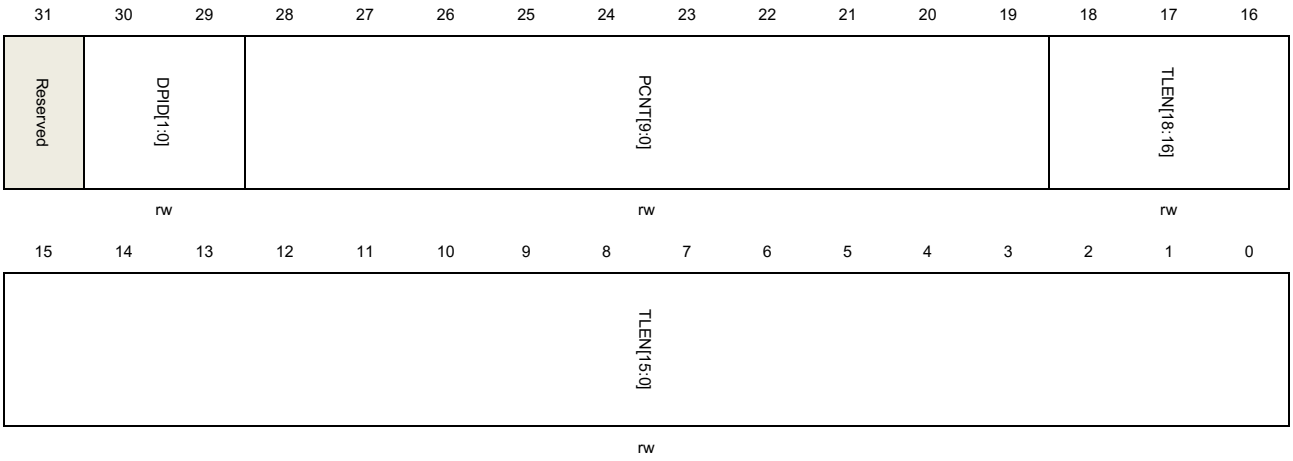
Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10	DTERIE	Data toggle error interrupt enable 0: Disable data toggle error interrupt 1: Enable data toggle error interrupt
9	REQOVRIE	Request queue overrun interrupt enable 0: Disable request queue overrun interrupt 1: Enable request queue overrun interrupt
8	BBERIE	Babble error interrupt enable 0: Disable babble error interrupt 1: Enable babble error interrupt
7	USBERIE	USB bus error interrupt enable 0: Disable USB bus error interrupt 1: Enable USB bus error interrupt
6	Reserved	Must be kept at reset value.

5	ACKIE	ACK interrupt enable 0: Disable ACK interrupt 1: Enable ACK interrupt
4	NAKIE	NAK interrupt enable 0: Disable NAK interrupt 1: Enable NAK interrupt
3	STALLIE	STALL interrupt enable 0: Disable STALL interrupt 1: Enable STALL interrupt
2	Reserved	Must be kept at reset value.
1	CHIE	Channel halted interrupt enable 0: Disable channel halted interrupt 1: Enable channel halted interrupt
0	TFIE	Transfer finished interrupt enable 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Host channel-x transfer length register (USBFS_HCHxLEN) (x = 0..7, 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	Reserved	Must be kept at reset value.
30:29	DPID[1:0]	Data PID Software should write this field before the transfer starts. For OUT transfers, this field controls the Data PID of the first transmitted packet. For IN transfers, 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, USBFS changes and toggles this field automatically following the USB protocol.
00: DATA0
10: DATA1
11: SETUP (For control transfer only)
01: Reserved

28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted (OUT) or received (IN) in a transfer. Software should program this field before the channel is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.
18:0	TLEN[18:0]	Transfer length The total data bytes` number of a transfer. For OUT transfers, 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 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.

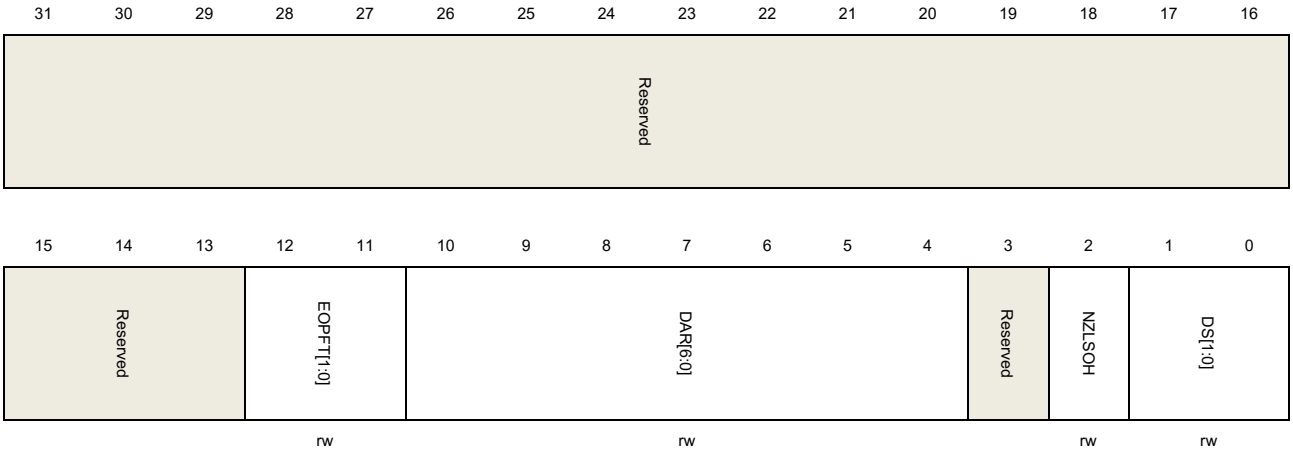
33.7.3. Device control and status registers

Device configuration register (USBFS_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)



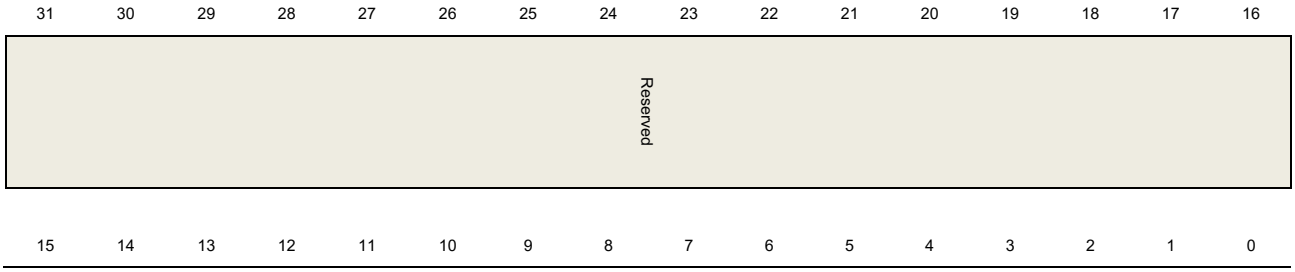
Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:11	EOPFT[1:0]	End of periodic frame time This field defines the percentage time point in a frame that the end of periodic frame (EOPF) flag should be triggered. 00: 80% of the frame time 01: 85% of the frame time 10: 90% of the frame time 11: 95% of the frame time
10:4	DAR[6:0]	Device address This field defines the USB device's address. USBFS 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 When a USB device receives a non-zero-length data packet during status OUT stage, this field controls that either USBFS 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 USBFS_DOEPxCTL register. 1: Send a STALL handshake and don't save the received OUT packet.
1:0	DS[1:0]	Device speed This field controls the device speed when the device connected to a host. 11: Full speed Others: Reserved

Device control register (USBFS_DCTL)

Address offset: 0x0804

Reset value: 0x0000 0000

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



Reserved	POIF	CGONAK	SGONAK	CGINAK	SGINAK	Reserved	GONS	GINS	SD	RWKUP
	rw	w	w	w	w		r	r	rw	rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	POIF	Power-on initialization finished Software should set this bit to notify USBFS that the registers are initialized after waking up from power down state.
10	CGONAK	Clear global OUT NAK Software sets this bit to clear GONS bit in this register.
9	SGONAK	Set global OUT NAK Software sets this bit to set GONS bit in this register. When GONS bit is zero, setting this bit will also cause GONAK flag in USBFS_GINTF register triggered after a while. Software should clear the GONAK flag before writing this bit again.
8	CGINAK	Clear global IN NAK Software sets this bit to clear GINS bit in this register.
7	SGINAK	Set global IN NAK Software sets this bit to set GINS bit in this register. When GINS bit is zero, setting this bit will also cause GINAK flag in USBFS_GINTF register triggered after a while. Software should clear the GINAK flag before writing this bit again.
6:4	Reserved	Must be kept at reset value.
3	GONS	Global OUT NAK status 0: The handshake that USBFS 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. 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 0: The response to IN transaction is decided by Tx FIFO status, endpoint's NAK and STALL bits. 1: USBFS always responses to IN transaction with a NAK handshake.
1	SD	Soft disconnect Software can use this bit to generate a soft disconnect condition on USB bus. After this bit is set, USBFS switches off the pull up resistor on DP line. This will cause the

host to detect a device disconnect.
0: No soft disconnect generated.
1: Generate a soft disconnection.

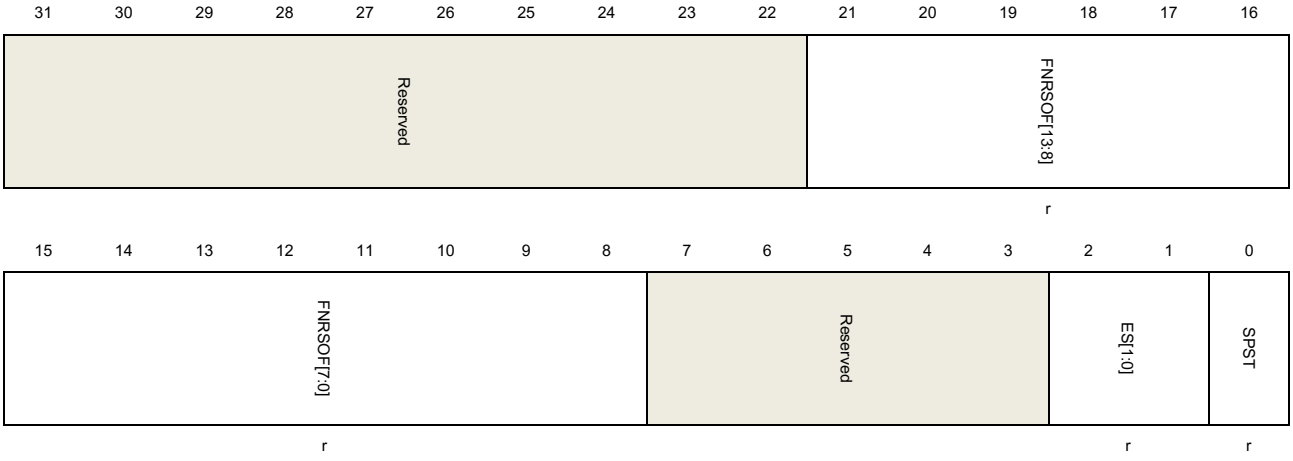
0 RWKUP Remote wakeup
In suspend state, software can use this bit to generate a Remote wake up signal to inform host that it should resume the USB bus.
0: No remote wakeup signal generated.
1: Generate remote wakeup signal.

Device status register (USBFS_DSTAT)

Address offset: 0x0808
Reset value: 0x0000 0000

This register contains status and information of the USBFS 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. USBFS 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 This field reports the enumerated device speed. Read this field after the ENUMF flag in USBFS_GINTF register is triggered. 11: Full speed Others: reserved
0	SPST	Suspend status This bit reports whether device is in suspend state. 0: Device is not in suspend state.

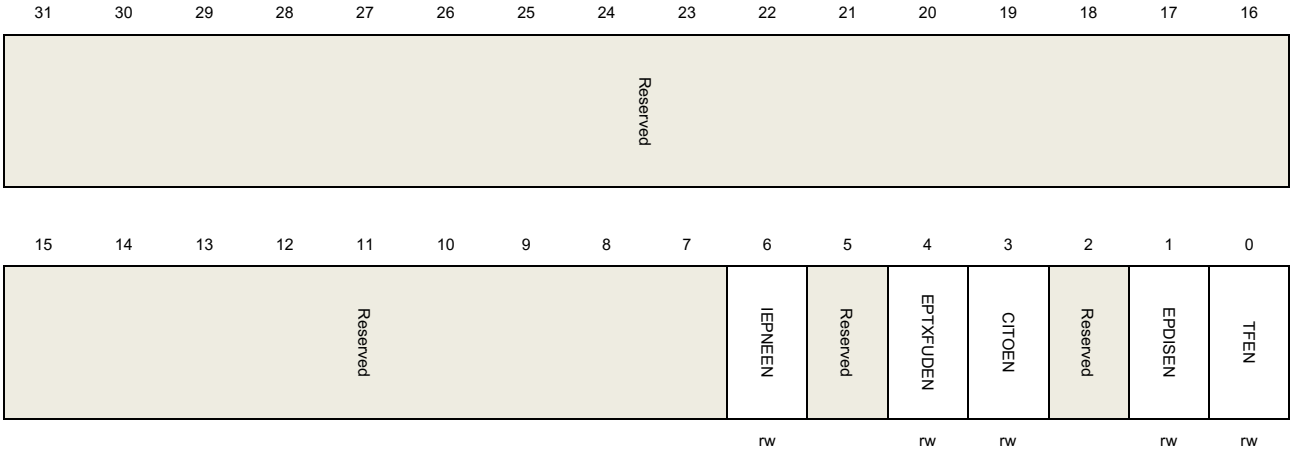
1: Device is in suspend state.

Device IN endpoint common interrupt enable register (USBFS_DIEPINTEN)

Address offset: 0x810
Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS_DIEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS_DIEPxINTF register is able to trigger an endpoint interrupt in USBFS_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
31:7	Reserved	Must be kept at reset value.
6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable IN endpoint NAK effective interrupt 1: Enable IN endpoint NAK effective interrupt
5	Reserved	Must be kept at reset value.
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable endpoint Tx FIFO underrun interrupt 1: Enable endpoint Tx FIFO underrun interrupt
3	CITOEN	Control IN timeout interrupt enable bit 0: Disable control IN timeout interrupt 1: Enable control IN timeout interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable endpoint disabled interrupt 1: Enable endpoint disabled interrupt

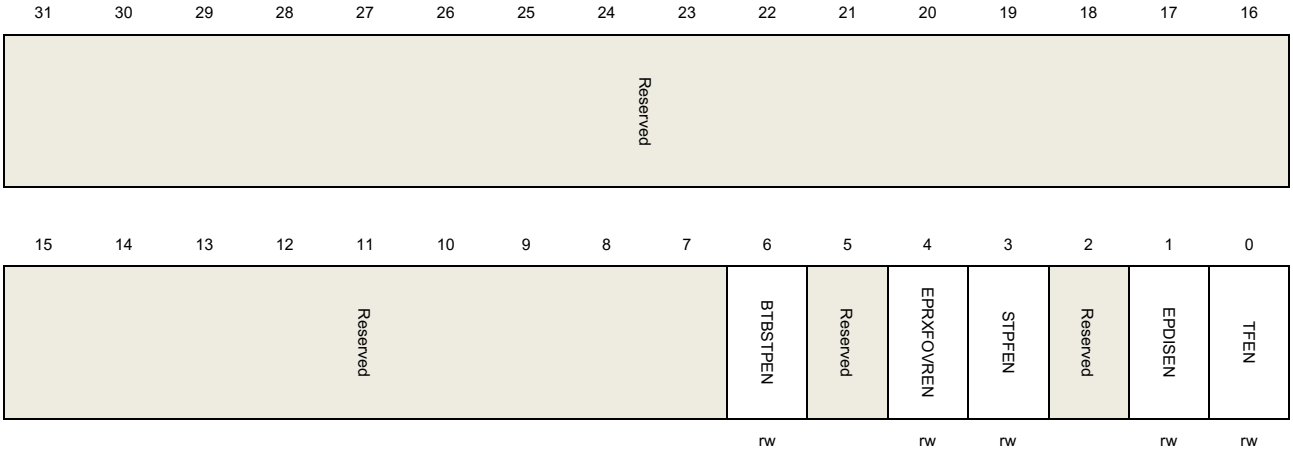
0	TFEN	Transfer finished interrupt enable bit 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Device OUT endpoint common interrupt enable register (USBFS_DOEPINTEN)

Address offset: 0x0814
Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS_DOEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS_DOEPxINTF register is able to trigger an endpoint interrupt in USBFS_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
31:7	Reserved	Must be kept at reset value.
6	BTBSTPEN	Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit 0: Disable back-to-back SETUP packets interrupt 1: Enable back-to-back SETUP packets interrupt
5	Reserved	Must be kept at reset value.
4	EPRXFOVREN	Endpoint Rx FIFO overrun interrupt enable bit 0: Disable endpoint Rx FIFO overrun interrupt 1: Enable endpoint Rx FIFO overrun interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable SETUP phase finished interrupt 1: Enable SETUP phase finished interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit

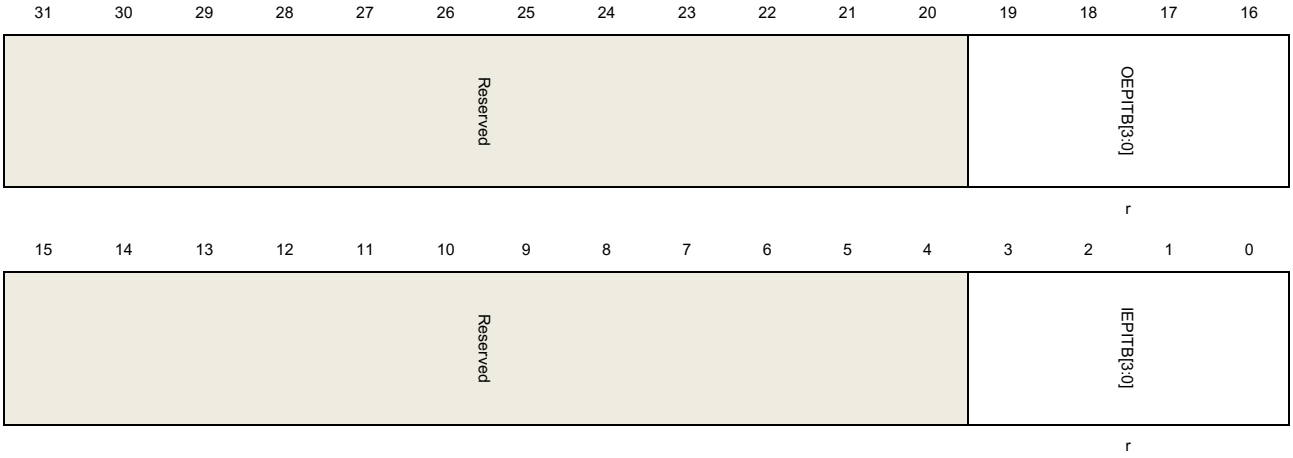
		0: Disable endpoint disabled interrupt 1: Enable endpoint disabled interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Device all endpoints interrupt register (USBFS_DAEPINT)

Address offset: 0x0818
Reset value: 0x0000 0000

When an endpoint interrupt is triggered, USBFS 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
31:20	Reserved	Must be kept at reset value.
19:16	OEPITB[3:0]	Device all OUT endpoint interrupt bits Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	Reserved	Must be kept at reset value.
3:0	IEPITB[3:0]	Device all IN endpoint interrupt bits Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3.

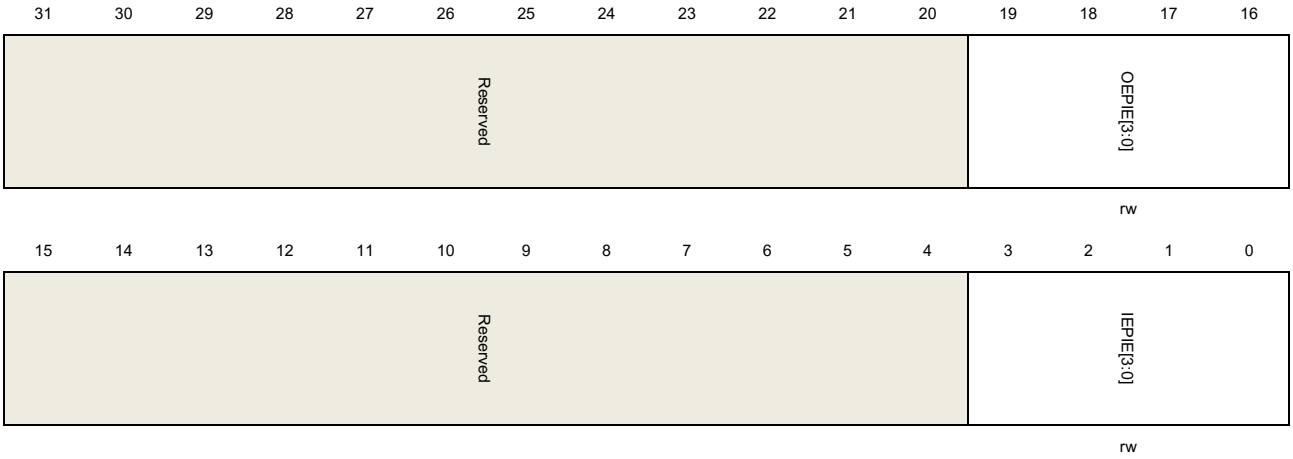
Device all endpoints interrupt enable register (USBFS_DAEPINTEN)

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 USBFS_GINTF register.

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



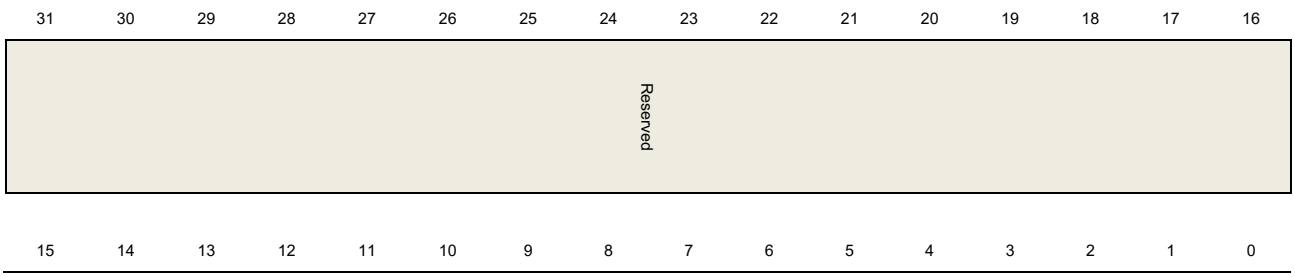
Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:16	OEPIE[3:0]	Out endpoint interrupt enable 0: Disable OUT endpoint-n interrupt 1: Enable OUT endpoint-n interrupt Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	Reserved	Must be kept at reset value.
3:0	IEPIE[3:0]	IN endpoint interrupt enable bits 0: Disable IN endpoint-n interrupt 1: Enable IN endpoint-n interrupt Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3.

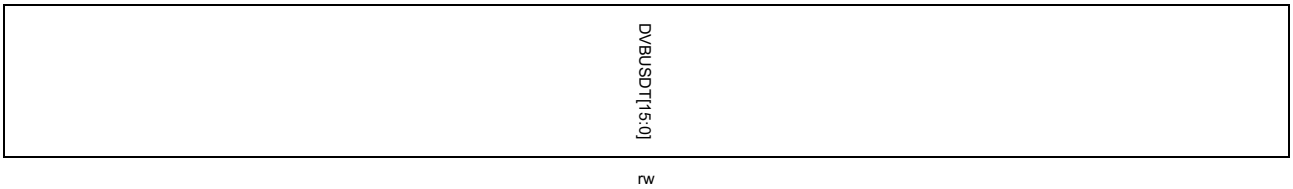
Device VBUS discharge time register (USBFS_DVBUSDT)

Address offset: 0x0828

Reset value: 0x0000 17D7

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



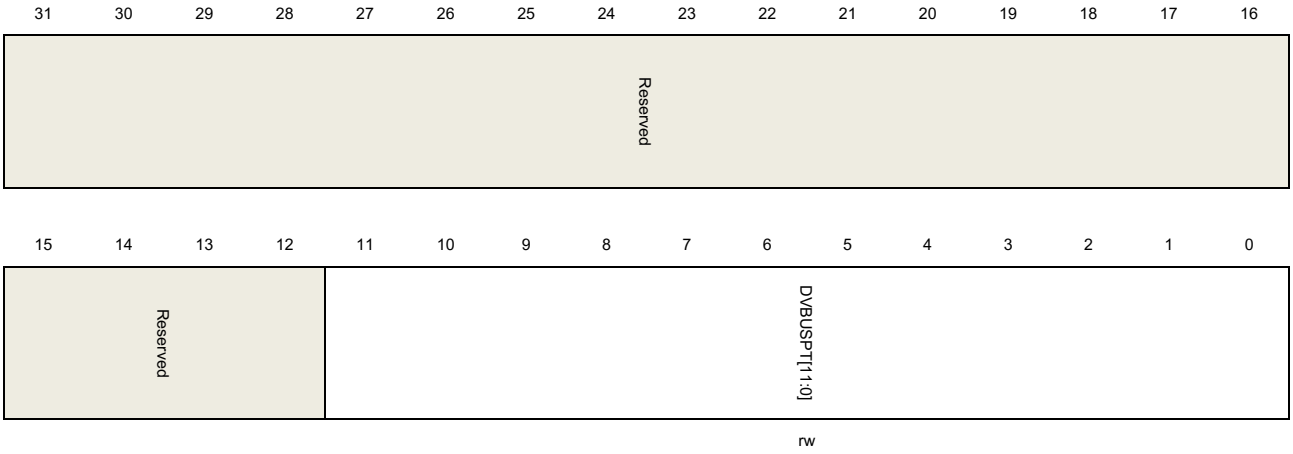


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	DVBUSDT[15:0]	Device V_{BUS} discharge time There is a discharge process after V_{BUS} pulsing in SRP protocol. This field defines the discharge time of V_{BUS}. The true discharge time is 1024 * DVBUSDT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device VBUS pulsing time register (USBFS_DVBUSPT)

Address offset: 0x082C
Reset value: 0x0000 05B8

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



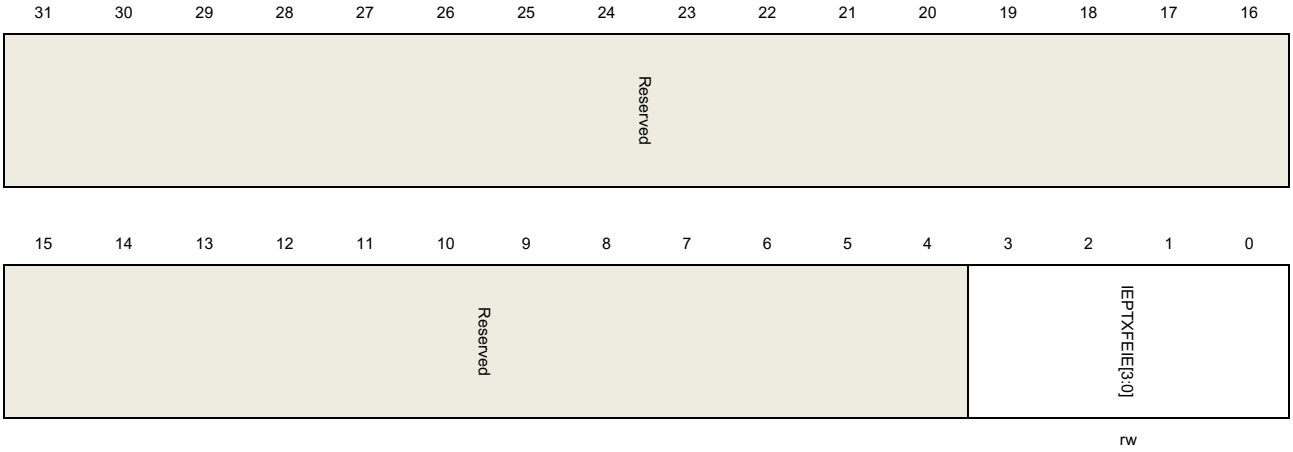
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	DVBUSPT[11:0]	Device V_{BUS} pulsing time This field defines the pulsing time for V_{BUS}. The true pulsing time is 1024*DVBUSPT[11:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device IN endpoint FIFO empty interrupt enable register (USBFS_DIEPFEINTEN)

Address offset: 0x0834
Reset value: 0x0000 0000

This register contains the enable bits for the Tx FIFO empty interrupts of IN endpoints.

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



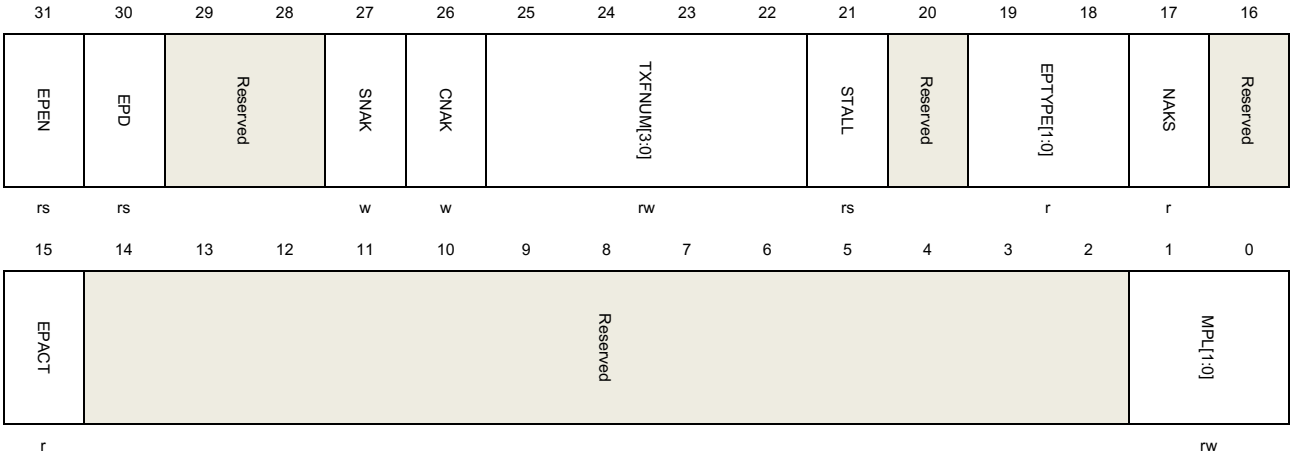
Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3:0	IEPTXFEIE[3:0]	IN endpoint Tx FIFO empty interrupt enable bits This field controls whether the TXFE bits in USBFS_DIEPxINTF registers are able to generate an endpoint interrupt bit in USBFS_DAEPINT register. Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3 0: Disable FIFO empty interrupt 1: Enable FIFO empty interrupt

Device IN endpoint 0 control register (USBFS_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 Set by the application and cleared by USBFS.

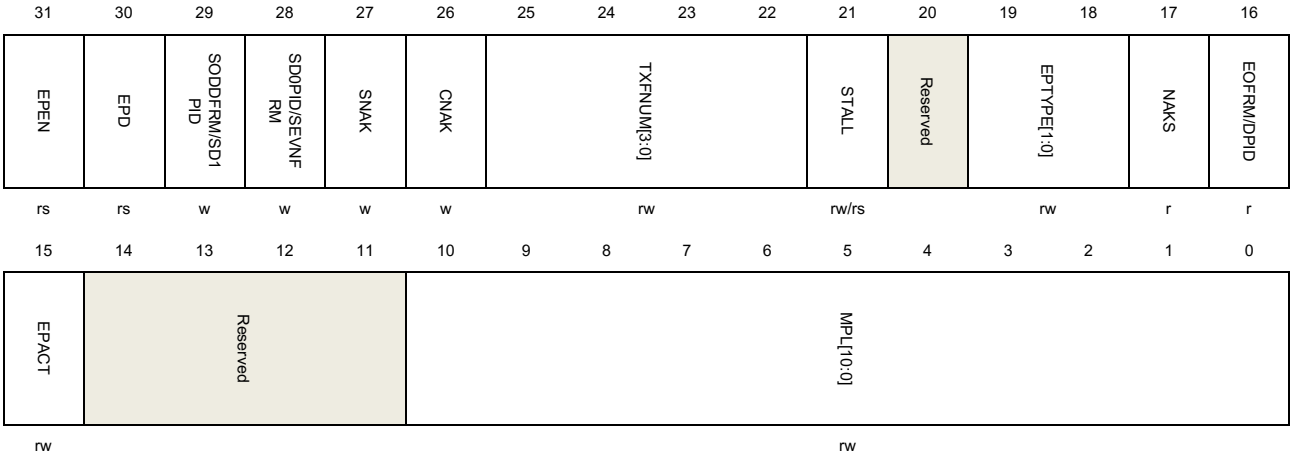
		0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should following the operation guide to disable or enable an endpoint.
29:28	Reserved	Must be kept at reset value.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Defines the Tx FIFO number of IN endpoint 0.
21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake when receiving IN token. USBFS 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 USBFS_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 This field is fixed to '00' for control endpoint.
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GINS bit in USBFS_DCTL register are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint's Tx FIFO. 1: USBFS always sends NAK handshake to the IN token. 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 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 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: 00: 64 bytes

01: 32 bytes
10: 16 bytes
11: 8 bytes

Device IN endpoint-x control register (USBFS_DIEPxCTL) (x = 1..3, 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 Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable 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) This bit has effect only if this is an isochronous IN endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk IN endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM	Set even frame (For isochronous IN endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk IN endpoints) Software sets this bit to clear DPID bit in this register.

27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Defines the Tx FIFO number of this IN endpoint.
21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake when receiving IN token. This bit has a higher priority than NAKS bit in this register and GINS bit in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect. For control IN endpoint: Only USBFS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it. For interrupt or bulk IN endpoint: Only software can clear this bit
20	Reserved	Must be kept at reset value.
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GINS bit in USBFS_DCTL register are are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint's Tx FIFO. 1: USBFS always sends NAK handshake to the IN token. 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) For isochronous transfers, software can use this bit to control that USBFS only sends data packets for IN tokens in even or odd frames. If the parity of the current frame number doesn't match with this bit, USBFS only responses with a zero-length packet. 0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk IN endpoints) There is a data PID toggle scheme in interrupt or bulk transfer. Set SD0PID to set

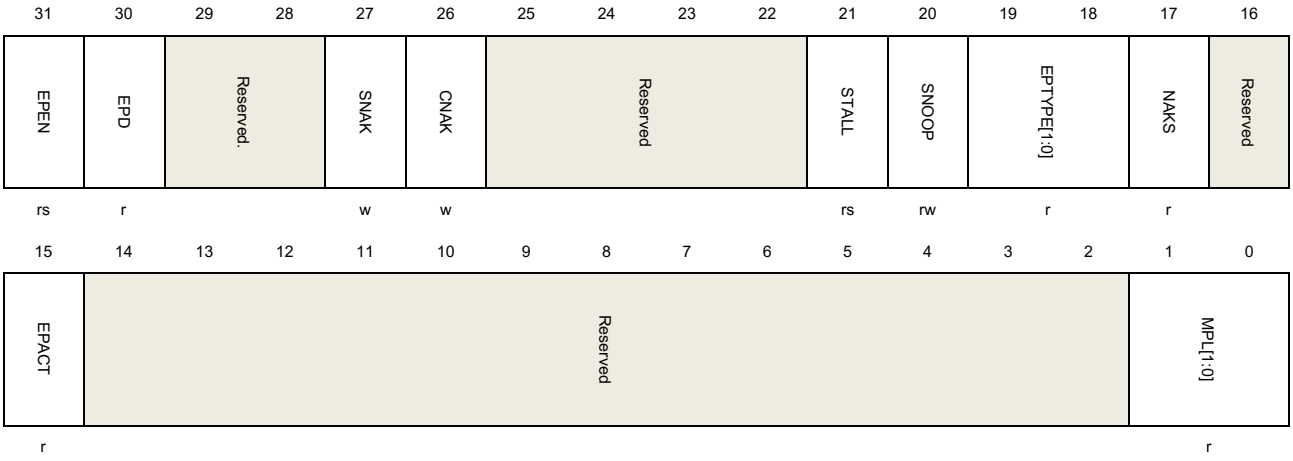
		this bit before a transfer starts and USBFS maintains this bit during transfers according to the data toggle scheme described in USB protocol. 0: Data packet's PID is DATA0 1: Data packet's PID is DATA1
15	EPACT	Endpoint active This bit controls whether this endpoint is active. If an endpoint is not 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 (USBFS_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 Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable This bit is fixed to 0 for OUT endpoint 0.
29:28	Reserved	Must be kept at reset value.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK

		Software sets this bit to clear NAKS bit in this register
25:22	Reserved	Must be kept at reset value.
21	STALL	STALL handshake Set this bit to make USBFS send STALL handshake during an OUT transaction. USBFS 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 This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value. 0: Snoop mode disabled 1: Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type This field is fixed to '00' for control endpoint.
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint's Rx FIFO. 1: USBFS always sends NAK handshake for the OUT token. 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 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 This is a read-only field, and its value comes from the MPL field of USBFS_DIEP0CTL register: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

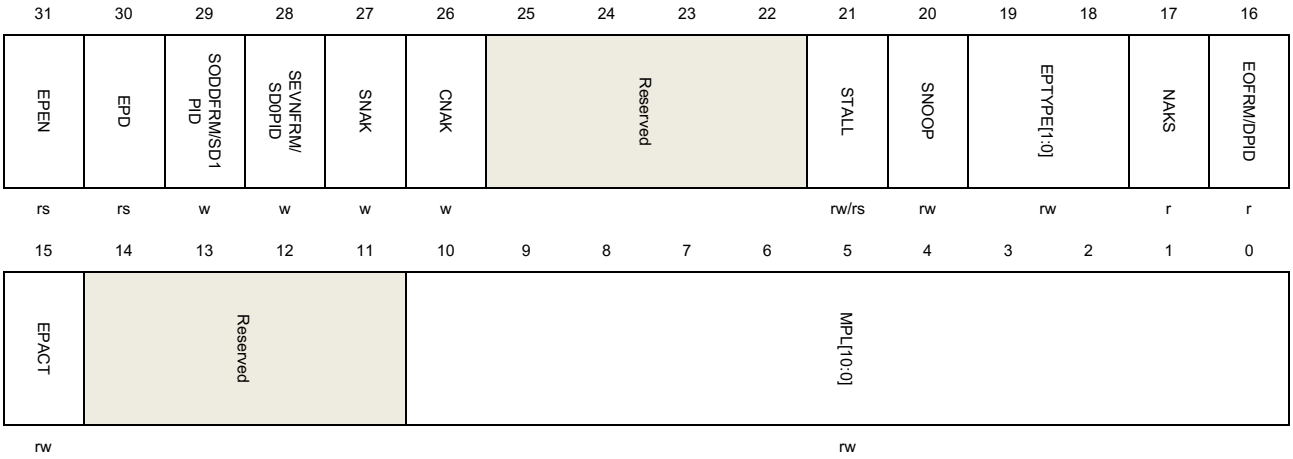
Device OUT endpoint-x control register (USBFS_DOEPxCTL) (x = 1..3, where x = endpoint_number)

Address offset: 0x0B00 + (endpoint_number × 0x20)

Reset value: 0x0000 0000

The application uses this register to control the operations of each logical OUT endpoint other than OUT endpoint 0.

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



Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable 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) This bit has effect only if this is an isochronous OUT endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk OUT endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM	Set even frame (For isochronous OUT endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk OUT endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	Reserved	Must be kept at reset value.

21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake during an OUT transaction. This bit has a higher priority than NAKS bit in this register and GINAK in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect. For control OUT endpoint: Only USBFS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it. For interrupt or bulk OUT endpoint: Only software can clear this bit.
20	SNOOP	Snoop mode This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value. 0: Snoop mode disabled 1: Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared: 0: USBFS sends handshake packets according to the status of the endpoint's Rx FIFO. 1: USBFS always sends NAK handshake to the OUT token. 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) For isochronous transfers, software can use this bit to control that USBFS only receives data packets in even or odd frames. If the current frame number's parity doesn't match with this bit, USBFS just drops the data packet. 0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk OUT endpoints) 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 USBFS maintains this bit during transfers following the data toggle scheme described in USB protocol. 0: Data packet's PID is DATA0 1: Data packet's PID is DATA1

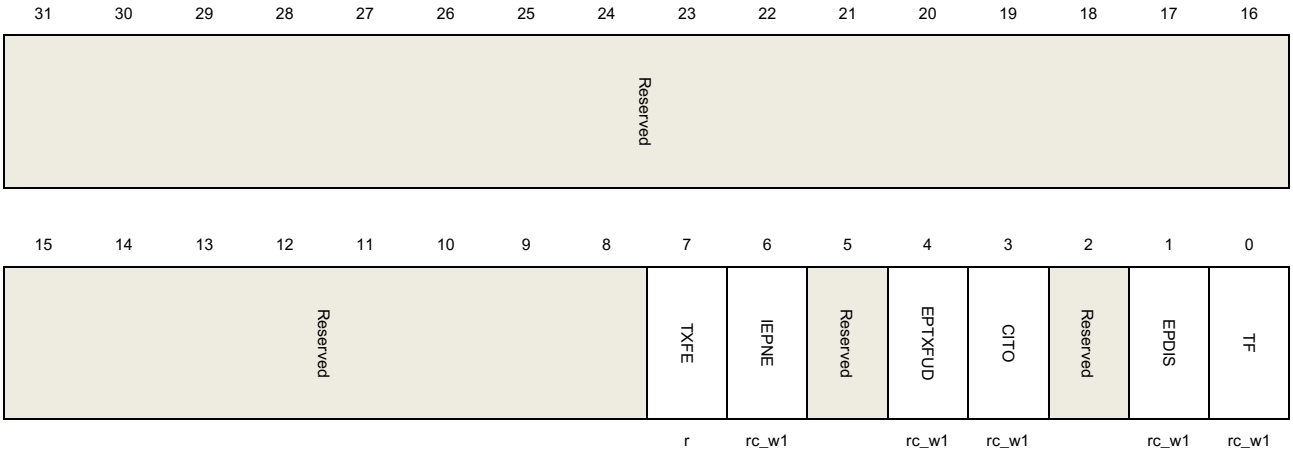
15	EPACT	Endpoint active This bit controls whether this endpoint is active. If an endpoint is not active, it ignores all tokens and doesn't make any response.
14:11	Reserved	Must be kept at reset value.
10:0	MPL[10:0]	This field defines the maximum packet length in bytes.

Device IN endpoint-x interrupt flag register (USBFS_DIEPxINTF) (x = 0..3, 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 an IN endpoint interrupt occurs, 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
31:8	Reserved	Must be kept at reset value.
7	TXFE	Transmit FIFO empty The Tx FIFO of this IN endpoint has reached the empty threshold value defined by TXFTH field in USBFS_GAHBCS register.
6	IEPNE	IN endpoint NAK effective The setting of SNAK bit in USBFS_DIEPxCTL register takes effect. This bit can be cleared either by writing 1 to it or by setting CNAK bit in USBFS_DIEPxCTL register.
5	Reserved	Must be kept at reset value.
4	EPTXFUD	Endpoint Tx FIFO underrun This flag is triggered if the Tx FIFO has no packet data when an IN token is

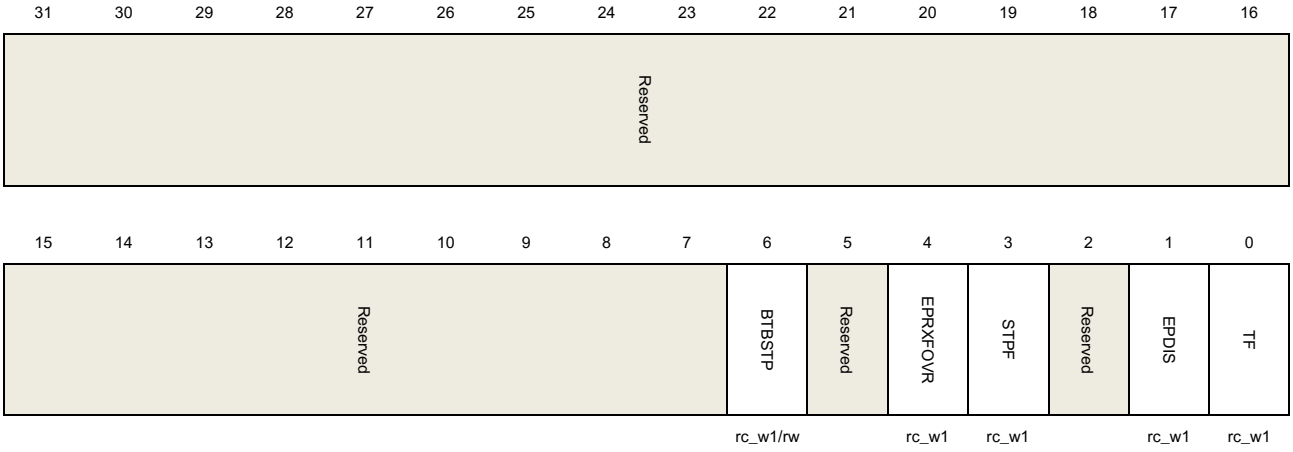
		incoming.
3	CITO	Control IN Timeout interrupt This flag is triggered if the device waiting for a handshake is timeout in a control IN transaction.
2	Reserved	Must be kept at reset value.
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software's request.
0	TF	Transfer finished This flag is triggered when all the IN transactions assigned to this endpoint have been finished.

Device OUT endpoint-x interrupt flag register (USBFS_DOEPxINTF) (x = 0..3, 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 an OUT endpoint interrupt occurs, 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)



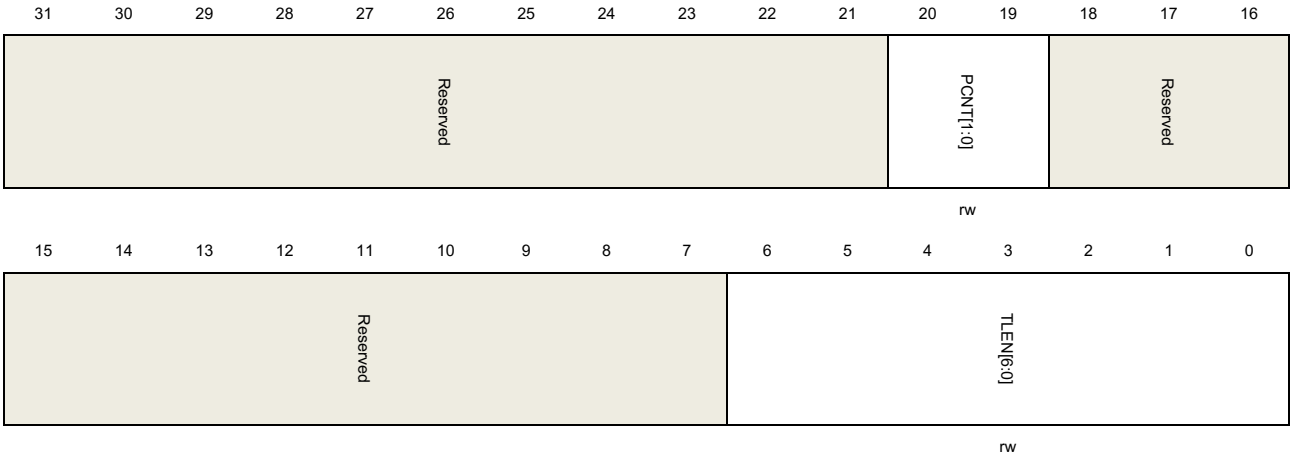
Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6	BTBSTP	Back-to-back SETUP packets (Only for control OUT endpoint) This flag is triggered when a control out endpoint has received more than 3 back-to-back setup packets.
5	Reserved	Must be kept at reset value.
4	EPRXFOVR	Endpoint Rx FIFO overrun

		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. USBFS will drop the incoming OUT data packet and sends 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. USBFS receives an IN or OUT token after a setup token.
2	Reserved	Must be kept at reset value.
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software's request.
0	TF	Transfer finished This flag is triggered when all the OUT transactions assigned to this endpoint have been finished.

Device IN endpoint 0 transfer length register (USBFS_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 The number of data packets desired to be transmitted in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.
18:7	Reserved	Must be kept at reset value.
6:0	TLEN[6:0]	Transfer length

The total data bytes` number of a transfer.

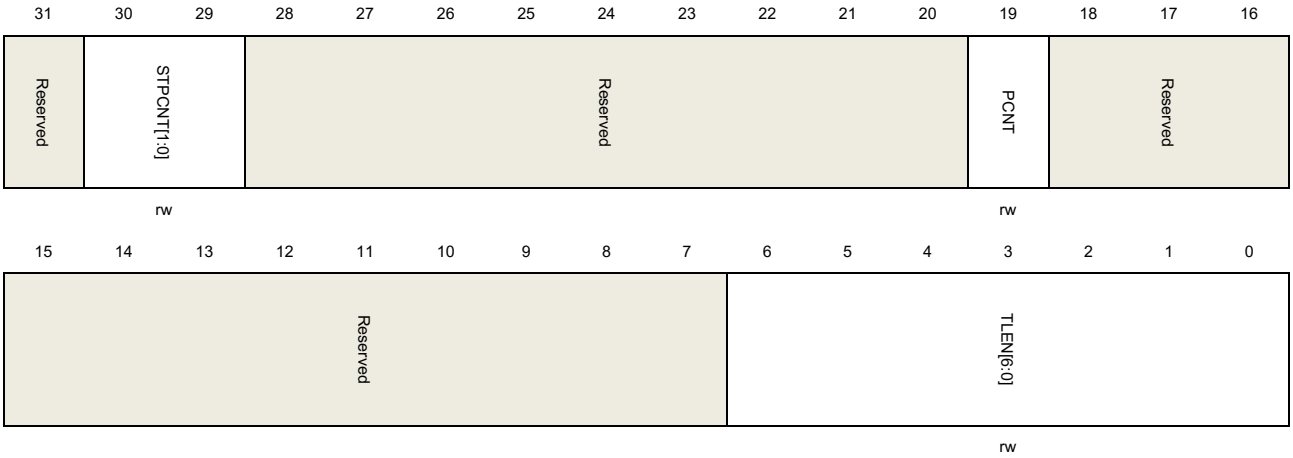
This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet.

Device OUT endpoint 0 transfer length register (USBFS_DOEP0LEN)

Address offset: 0x0B10

Reset value: 0x0000 0000

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



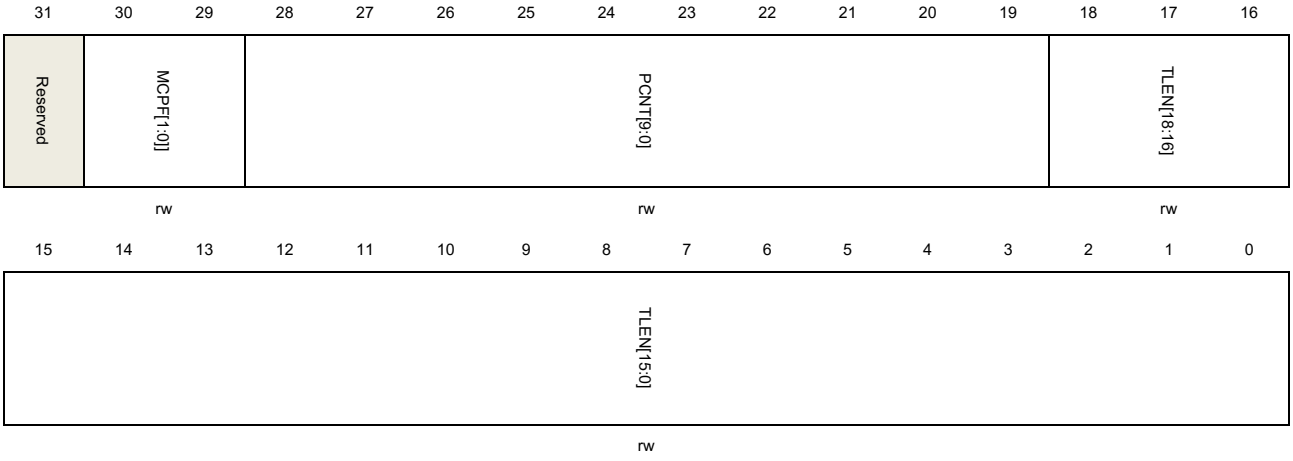
Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	STPCNT[1:0]	SETUP packet count This field defines the maximum number of back-to-back SETUP packets this endpoint can accept. Program this field before setup transfers. Each time a back-to-back setup packet is received, USBFS decrease this field by one. When this field reaches zero, the BTBSTP flag in USBFS_DOEP0INTF register will be triggered. 00: 0 packet 01: 1 packet 10: 2 packets 11: 3 packets
28:20	Reserved	Must be kept at reset value.
19	PCNT	Packet count The number of data packets desired to receive in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet reception on bus.

18:7	Reserved	Must be kept at reset value.
6:0	TLEN[6:0]	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. Program this field before the endpoint is enabled. Each time software 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 (USBFS_DIEPxLEN) (x = 1 .. 3, where x = endpoint_number)

Address offset: 0x910 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

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



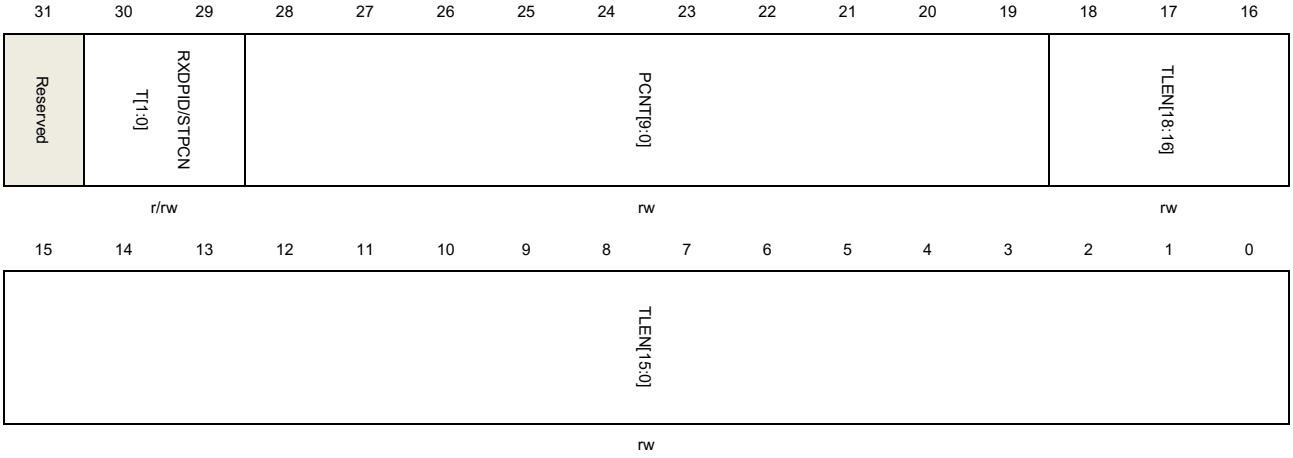
Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	MCPF[1:0]	Multi packet count per frame This field indicates the packet count that must be transmitted per frame for periodic IN endpoints on the USB. It is used to calculate the data PID for isochronous IN endpoints by the core. 01: 1 packet 10: 2 packets 11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.

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 be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software 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 (USBFS_DOEPxLEN) (x = 1 .. 3, 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) This field saves the PID of the latest received data packet on this endpoint. 00: DATA0 10: DATA1 Others: Reserved
	STPCNT[1:0]	SETUP packet count (For control OUT Endpoints.) This field defines the maximum number of back-to-back SETUP packets this endpoint can accept. Program this field before setup transfers. Each time a back-to-back setup packet is received, USBFS decrease this field by one. When this field reaches zero, the BTBSTP flag in USBFS_DOEPxINTF register will be triggered. 00: 0 packet 01: 1 packet 10: 2 packets

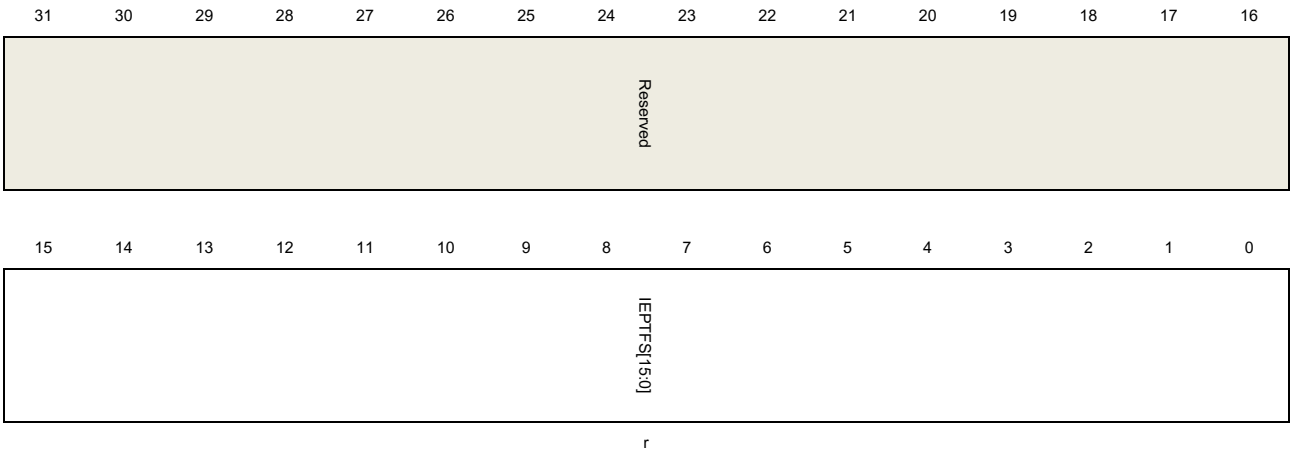
		11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to receive in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet reception on bus.
18:0	TLEN[18:0]	Transfer 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. Program this field before the endpoint is enabled. Each time after software reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

Device IN endpoint-x transmit FIFO status register (USBFS_DIEPxTFSTAT) (x = 0 .. 3, 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)



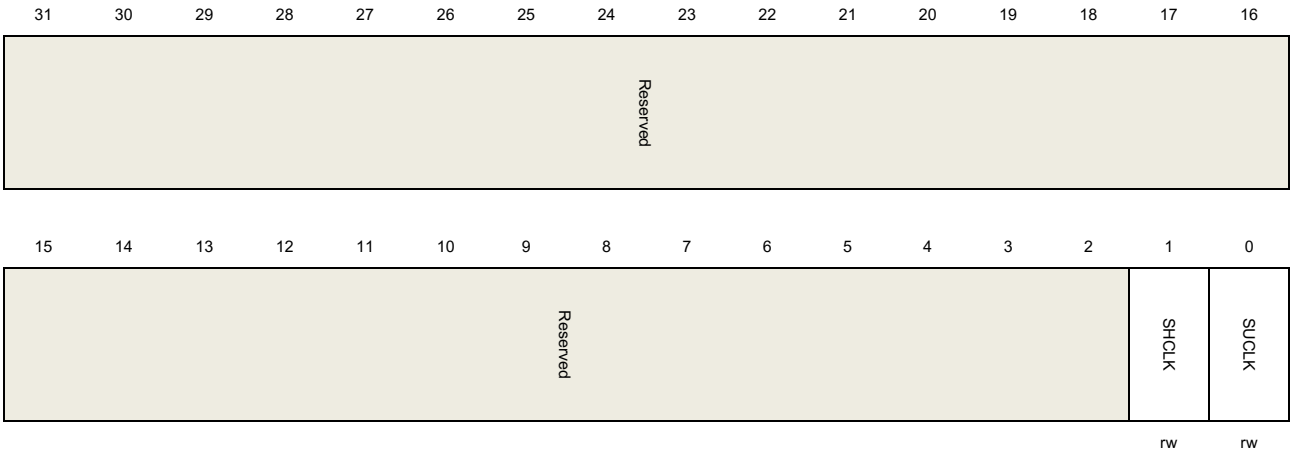
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	IEPTFS[15:0]	IN endpoint's Tx FIFO space remaining IN endpoint's Tx FIFO space remaining in 32-bit words: 0: FIFO is full 1: 1 word available … n: n words available

33.7.4. Power and clock control register (USBFS_PWRCLKCTL)

Address offset: 0x0E00

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	SHCLK	Stop HCLK Stop the HCLK to save power. 0: HCLK is not stopped 1: HCLK is stopped
0	SUCLK	Stop the USB clock Stop the USB clock to save power. 0: USB clock is not stopped 1: USB clock is stopped

34. Universal serial bus high-speed interface (USBHS)

34.1. 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 provides ULPI interface for external USB PHY integration and it also contains a full-speed USB PHY internal. So for full-speed or low-speed operation, no more external PHY chip is needed. USBHS supports all the four types of transfer (control, bulk, Interrupt and isochronous) defined in USB 2.0 protocol. HUB connection is supported when USBHS operates at high-speed in host mode. There is also a DMA engine operating as an AHB bus master in USBHS to speed up the data transfer between USBHS and system.

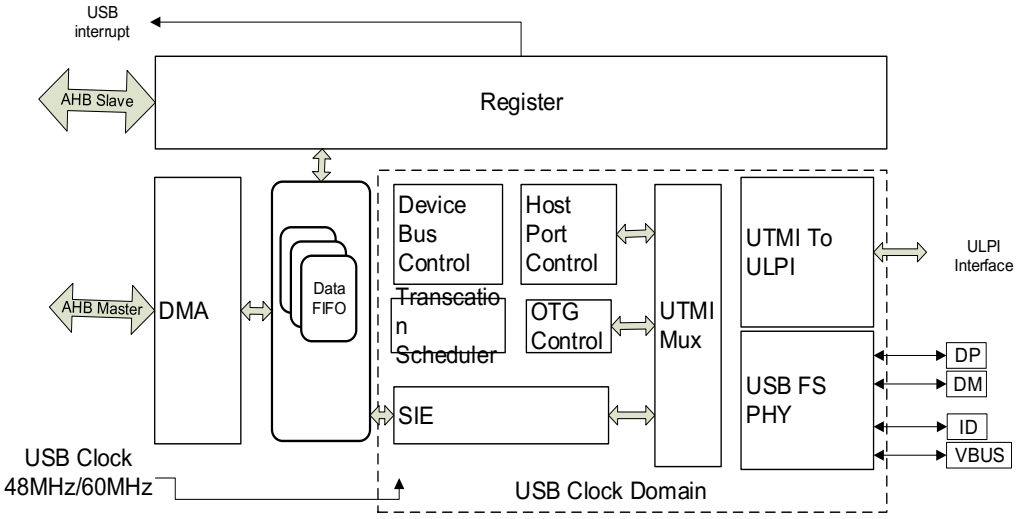
34.2. Characteristics

- 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 12 channels in host mode.
- Contains 2 transmit FIFOs (periodic and non-periodic) and one receive FIFO (shared by all channels) in host mode
- Contains 6 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 HUB connection in High-Speed host mode
- Supports 6 OUT and 6 IN Endpoints in device mode
- Supports remote-wakeup in device mode.
- Include a Full-Speed 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.

34.3. Block diagram

Figure 34-1. USBHS block diagram



34.4. Signal description

Table 34-1. USBHS signal description

IO port	Type	Description	Note
VBUS	Input	Bus power port	For internal PHY only
DM	Input / Output	Differential D-	For internal PHY only
DP	Input / Output	Differential D+	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
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

34.5. Function overview

34.5.1. 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 full-speed PHY and external ULPI PHY. The application chose to use either the internal embedded Full-Speed PHY or the external ULPI PHY according to the demand.

As shown in ***Table 34-2. USBHS supported speeds***, with internal PHY, the maximum speed supported by USBHS is full-speed, while using an external high-speed ULPI PHY, USBHS supports high-speed. The application may also limit the maximum speed of 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 34-2. USBHS supported speeds

Register configuration		Host supported speed	Device support speed
EMBPHY=1 (Internal PHY)		Full-Speed Low-Speed	Full-Speed
EMBPHY=0 (External ULPI PHY)	DS =01 (device mode) SPDFSLS=1 (host mode)	Full-Speed Low-Speed	Full-Speed
	DS =00 (device mode) SPDFSLS=0 (host mode)	High-Speed Full-Speed Low-Speed	High-Speed Full-Speed

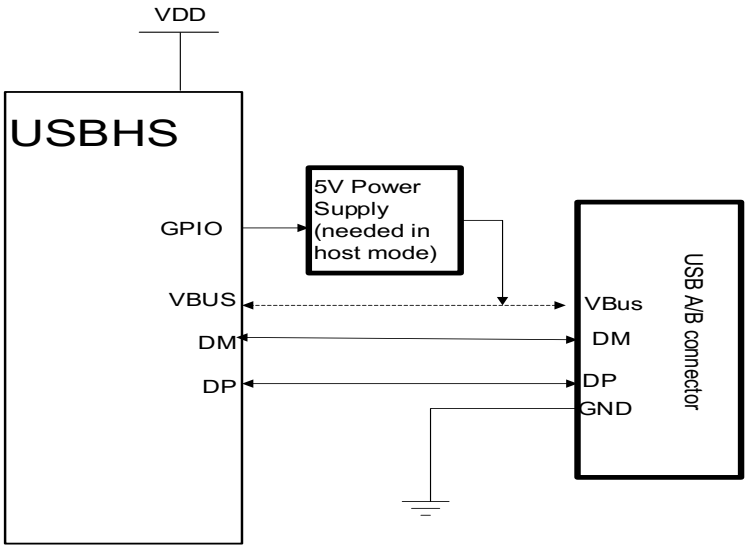
The application control the working modes of USBHS: force host，force device by set FHM and FDM bits in USBHS_GUSBCS register. When both bits are cleared, USBHS works in OTG mode, which is the default mode after system reset.

Internal Full-Speed PHY

USBHS includes an internal Full-Speed PHY. The internal PHY supports Full-Speed and Low-Speed in host mode, and Full-speed in device mode, supports OTG protocol with HNP and SRP. Software needs to set EMBPHY bit in USBHS_GUSBCS register to use this PHY. If internal full-speed PHY is selected, the USB clock used for the USBHS needs to be 48MHz. This 48MHz USB clock is generated from internal clocks in system, and its source and divider factors are configurable in RCU.

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 34-2. Connection using internal embedded PHY with host or device mode.***

Figure 34-2. Connection using internal embedded PHY with host or device mode

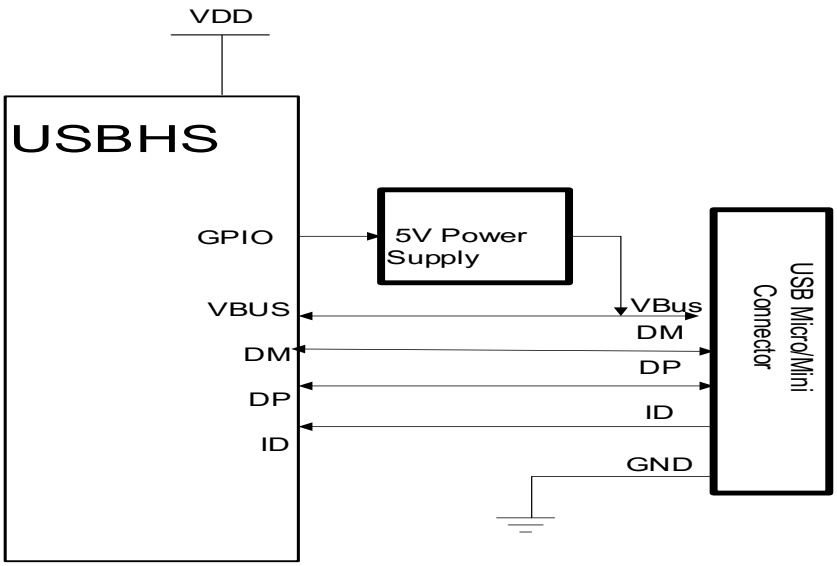


When USBHS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power pin defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuit on VBUS line. If application needs to supply USB power, an external 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 VBUSIG bit in USBHS_GCCFG register. If the device does not need to detect the voltage on VBUS pin, it may set the VBUSIG 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 34-3. Connection using internal embedded PHY with OTG mode***. When USBHS works in OTG mode, the FHM, FDM bits in USBHS_GUSBCS and VBUSIG 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 34-3. Connection using internal embedded PHY with OTG mode

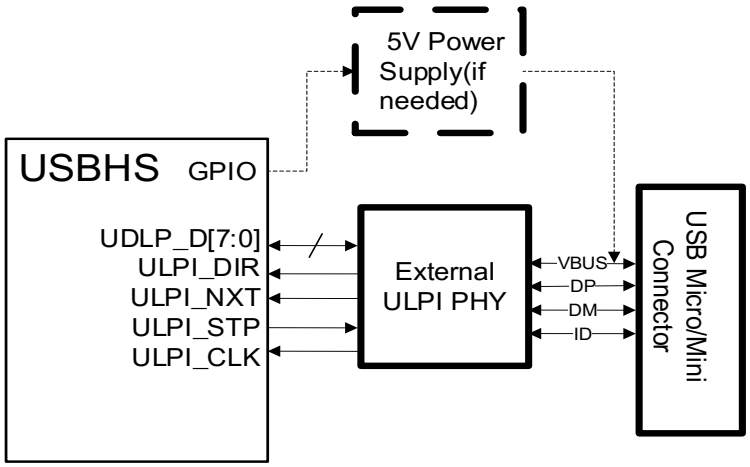


External ULPI PHY

As shown in ***Figure 34-4. Connection using 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 Full-Speed PHY.

Software needs to clear the EMBPHY 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 34-4. Connection using external ULPI PHY

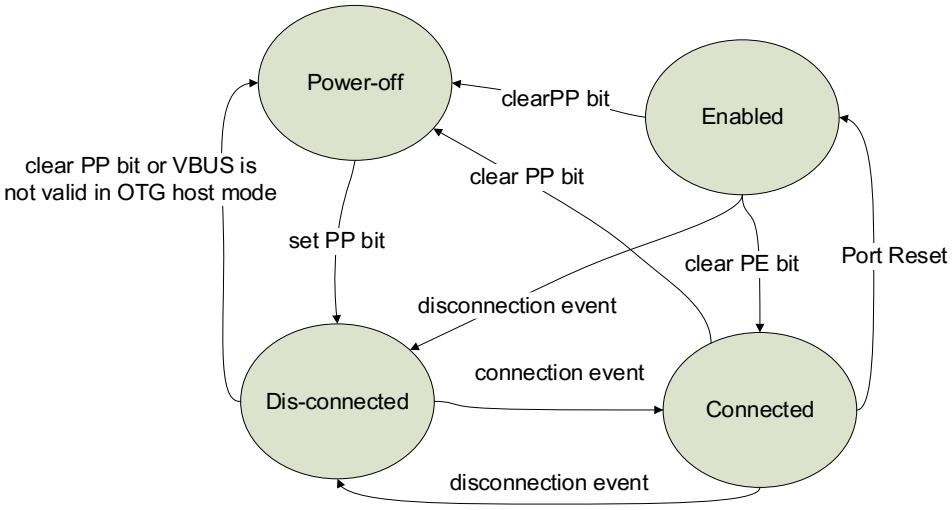


34.5.2. USB host function

USB Host Port State

Host application may control state of the USB port via USBHS_HPCS register. As shown in *Figure 34-5. State transition diagram of host port*, 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 34-5. 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[15:0] 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 16 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 12 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 kinds: 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.

34.5.3. 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 VBUSIG 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.

34.5.4. OTG function overview

USBHS supports OTG function described in OTG protocol 1.3; 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 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.

34.5.5. Data FIFO

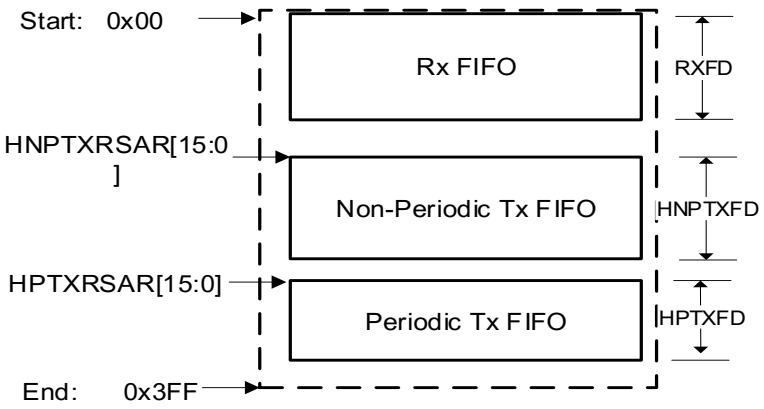
The USBHS include a 4K bytes data FIFO to store packet data. The data FIFO is implemented by using an internal SRAM in USBHS.

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. The ***Figure 34-6. HOST mode FIFO space in SRAM*** describes the structure of these FIFOs in SRAM. The values in the figure are in term

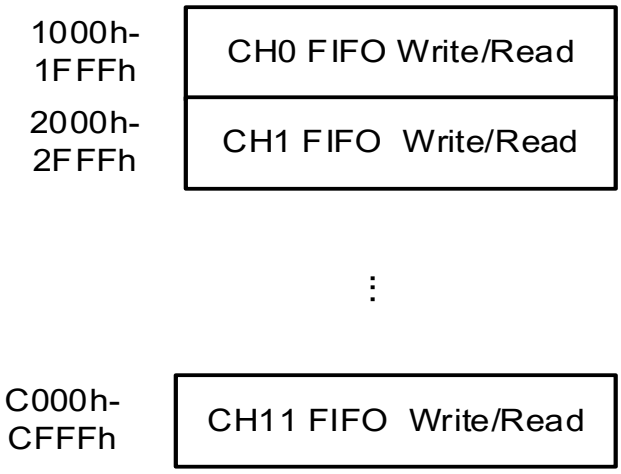
of 32-bit words.

Figure 34-6. 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. The ***Figure 34-7. 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 34-7. Host mode FIFO access register map

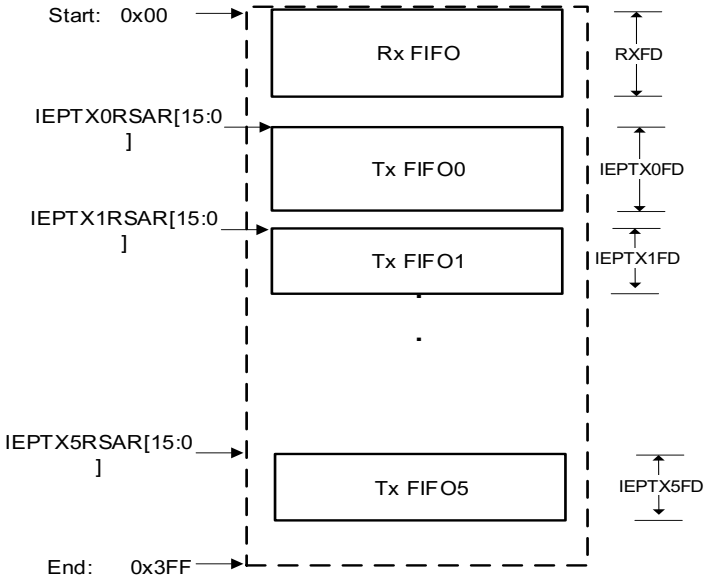


Device mode

In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 6 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…5) registers. The ***Figure 34-8. Device***

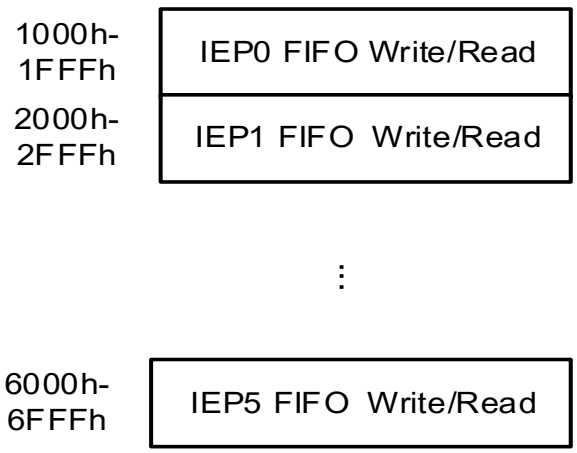
mode FIFO space in SRAM describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

Figure 34-8. 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. The *Figure 34-9. 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 34-9. Device mode FIFO access register map



34.5.6. 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 form 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.

34.5.7. 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 byte 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.

Note: Before starting to write the transmit FIFO, it is necessary to ensure that there is remaining space in the transmit FIFO and the request queue.

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 byte 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.

34.6. Interrupts

USBHS has four interrupts: global interrupt, wake-up interrupt, endpoint 1 IN interrupt and endpoint 1 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 ***Table 34-3. USBHS global interrupt.***

Table 34-3. USBHS global interrupt

Interrupt Flag	Description	Operation Mode
SESIF	Session interrupt	Host or device mode
DISCIF	Disconnect interrupt flag	Host Mode
IDPSC	ID pin status change	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
PXNCIF / ISOONCIF	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

Interrupt Flag	Description	Operation 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.

34.7. Register definition

USBHS start address: 0x4004 0000

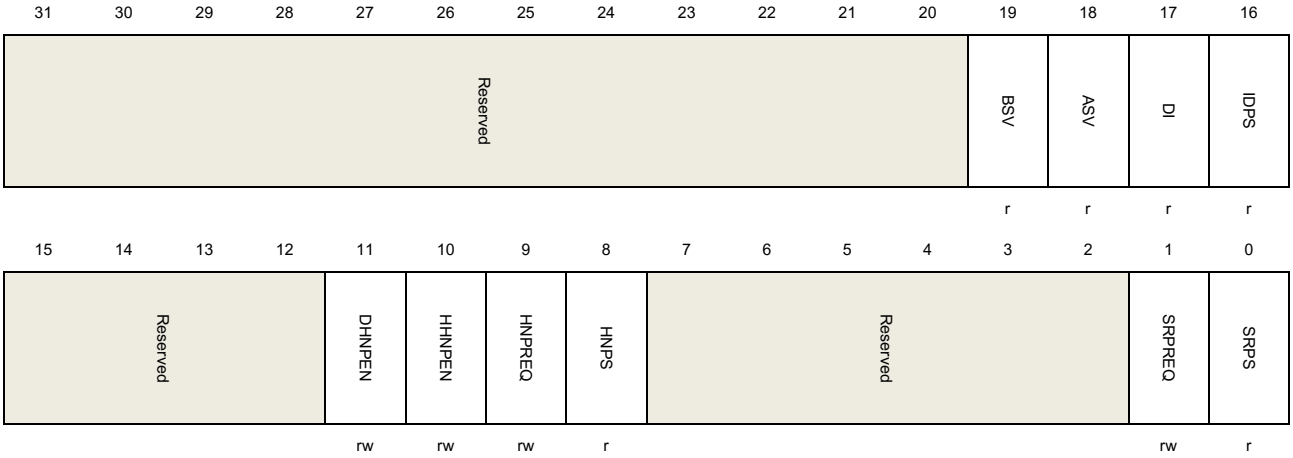
34.7.1. USBHS global registers

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
31:20	Reserved	Must be kept at reset value.
19	BSV	B-Session Valid (described in OTG protocol). 0: Vbus voltage level of a OTG B-device is below VBSESSVLD 1: Vbus voltage level of a OTG B-Device is not below VBSESSVLD **Note**: Only accessible in OTG B-Device mode.
18	ASV	A- session valid A-host mode transceiver status. 0: Vbus voltage level of a OTG A-device is below VASESSVLD 1: Vbus voltage level of a OTG A-device is below VASESSVLD The A-device is as host default at the start of a session. **Note**: Only accessible in OTG A-Device mode.
17	DI	Debounce interval Debounce interval of a detected connection. 0: Indicates the long debounce interval, when a plug-on and connection occur on USB bus 1: Indicates the short debounce interval, when a soft connection is used in HNP protocol.

		Note: Only accessible in host mode.
16	IDPS	ID pin status Voltage level of connector ID pin 0: USBHS is in A-device mode 1: USBHS is in B-device mode **Note**: Accessible in both device and host modes.
15:12	Reserved	Must be kept at reset value.
11	DHNPEN	Device HNP enable 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. 0: HNP function is not enabled. 1: HNP function is enabled **Note**: Only accessible in device mode.
10	HHNPEN	Host HNP enable Enable the HNP function of an A-device. If this bit is cleared, USBHS doesn't response to the HNP request from B-device. 0: HNP function is not enabled. 1: HNP function is enabled **Note**: Only accessible in host mode.
9	HNPREQ	HNP request 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. 0: Don't send HNP request 1: Send HNP request **Note**: Only accessible in device mode.
8	HNPS	HNP successes This bit is set by the core when HNP successes and cleared when HNPREQ bit is set. 0: HNP fails 1: HNP successes **Note**: Only accessible in device mode.
7:2	Reserved	Must be kept at reset value.
1	SRPREQ	SRP request 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. 0: No session request 1: Session request **Note**: Only accessible in device mode.

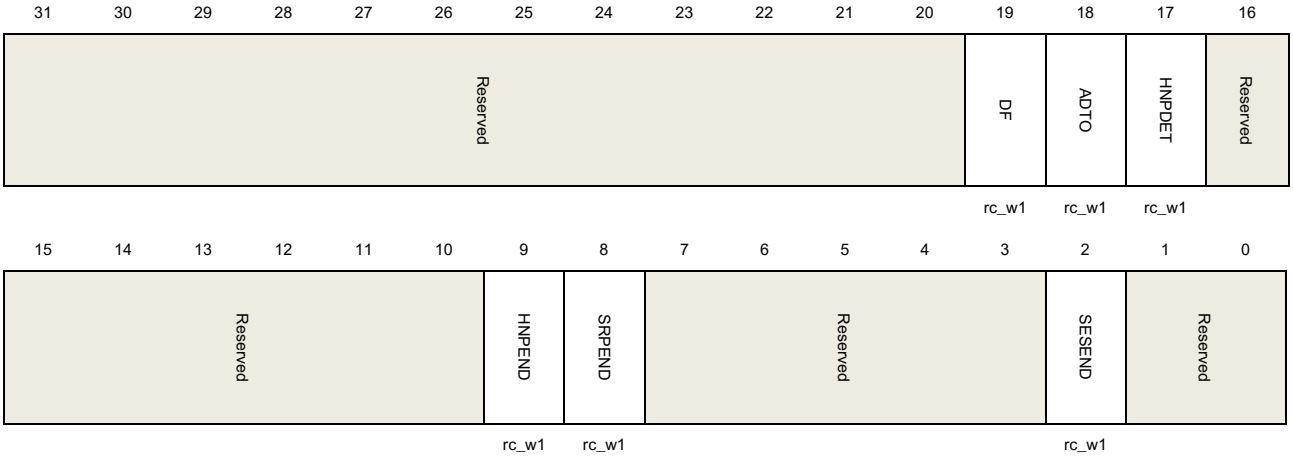
0	SRPS	SRP success This bit is set by the core when SRP successes and cleared when SRPREQ bit is set. 0: SRP fails 1: SRP successes **Note**: Only accessible in device mode.

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
31:20	Reserved	Must be kept at reset value.
19	DF	Debounce finish Set by USBHS when the debounce during device connection is done. **Note**: Only accessible in host mode.
18	ADTO	A-device timeout Set by USBHS when the A-device's waiting for a B-device's connection has timed out. **Note**: Accessible in both device and host modes.
17	HNPDET	Host negotiation request detected Set by USBHS when A-device detects a HNP request. **Note**: Accessible in both device and host modes.
16:10	Reserved	Must be kept at reset value.
9	HNPEND	HNP end Set by the core when a HNP ends. Software should read the HNPS in USBHS_GOTGCS register to get the result of HNP.

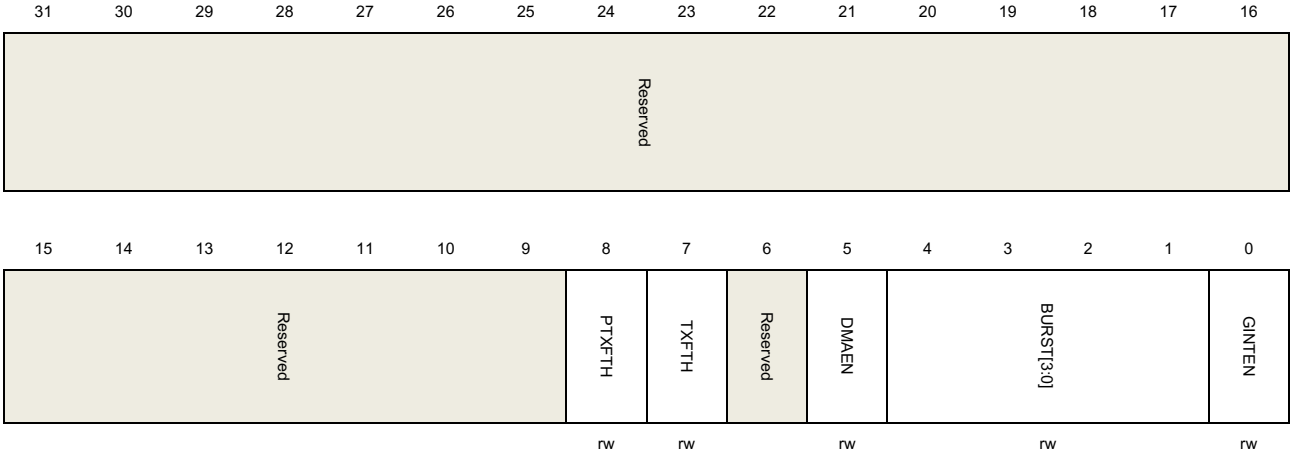
		Note: Accessible in both device and host modes.
8	SRPEND	SRPEND Set by the core when a SRP ends. Software should read the SRPS in USBHS_GOTGCS register to get the result of SRP. **Note**: Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value.
2	SESEND	Session end Set by the 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 0: PTXFEIF will be triggered when the periodic transmit FIFO is half empty 1: PTXFEIF will be triggered when the periodic transmit FIFO is completely empty **Note**: Only accessible in host mode.
7	TXFTH	Tx FIFO threshold Device mode: 0: TXFEIF will be triggered when the IN endpoint transmit FIFO is half empty 1: TXFEIF will be triggered when the IN endpoint transmit FIFO is completely empty Host mode: 0: NPTXFEIF will be triggered when the non-periodic transmit FIFO is half empty

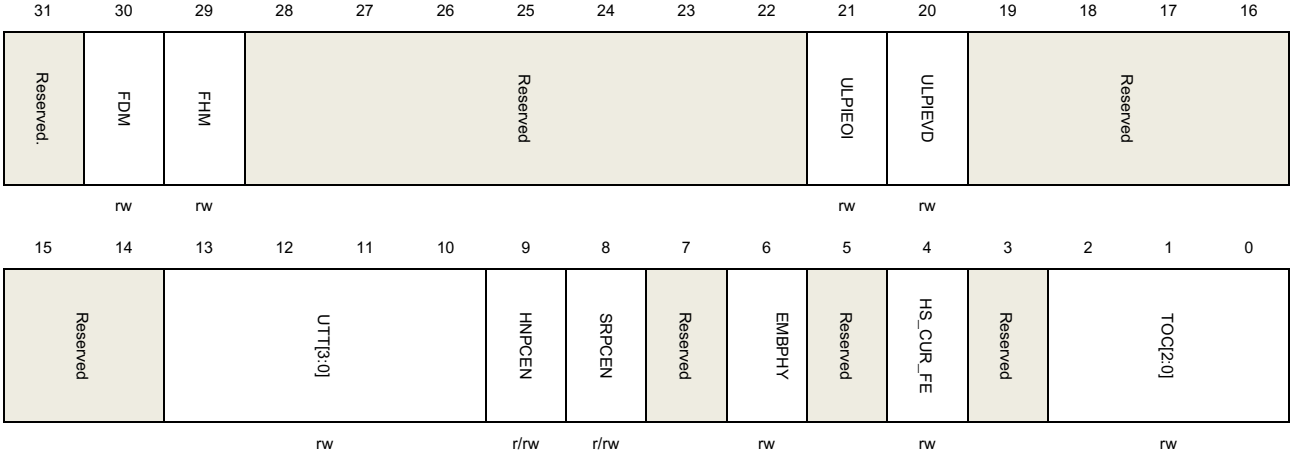
		1: NPTXFEIF will be triggered when the non-periodic transmit FIFO is completely empty
6	Reserved	Must be kept at reset value.
5	DMAEN	DMA function Enable 0: DMA function is disabled 1: DMA function is enabled
4:1	BURST[3:0]	The AHB burst type used by DMA 0000: Single 0001: INCR 0011: INCR4 0101: INCR8 0111: INCR16
0	GINTEN	Global interrupt enable 0: Global interrupt is not enabled. 1: Global interrupt is enabled. **Note**: Accessible in both device and host modes.

Global USB control and status register (USBHS_GUSBCS)

Address offset: 0x000C

Reset value: 0x0000 0A00

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 Setting this bit will force the core to device mode irrespective of the USBHS ID input pin. 0: Normal mode 1: Device mode

		The application must wait at least 25 ms for the change taking effect after setting the force bit. **Note**: Accessible in both device and host modes.
29	FHM	Force host mode Setting this bit will force the core to host mode irrespective of the USBHS ID input pin. 0: Normal mode 1: Host mode The application must wait at least 25 ms for the change taking effect after setting the force bit. **Note**: Accessible in both device and host modes.
28:22	Reserved	Must be kept at reset value.
21	ULPIEOI	ULPI external over-current indicator 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 bit in this register is 0). 0: ULPI PHY uses internal over-current indicator 1: ULPI PHY uses external over-current indicator
20	ULPIEVD	ULPI external VBUS driver 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 bit in this register is 0). 0: VBUS is driven by ULPI PHY 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 Turnaround time in PHY clocks. **Note**: Only accessible in device mode.
9	HNPCEN	HNP capability enable Controls whether the HNP capability is enabled 0: HNP capability is disabled 1: HNP capability is enabled **Note**: Accessible in both device and host modes.
8	SRPCEN	SRP capability enable Controls whether the SRP capability is enabled 0: SRP capability is disabled 1: SRP capability is enabled **Note**: Accessible in both device and host modes.
7	Reserved	Must be kept at reset value.

6	EMBPHY	Embedded PHY selected Controls whether the internal embedded Full-Speed PHY or the external ULPI PHY is used. 0: USBHS uses external ULPI PHY 1: USBHS uses the internal embedded Full-Speed PHY **Note**: Accessible in both device and host modes.
5	Reserved	Must be kept at reset value.
4	HS_CUR_FE	HS current software enable 0: Release the HS mode TX current enable 1: Force HS mode TX current enable
3	Reserved	Must be kept at reset value.
2:0	TOC[2:0]	Timeout calibration 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.)

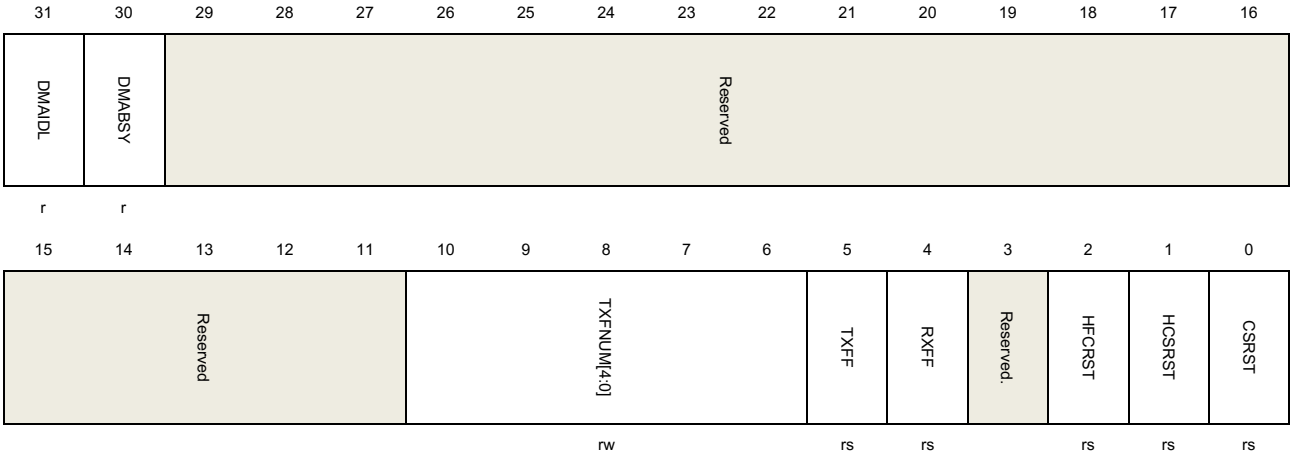
Global reset control register (USBHS_GRSTCTL)

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 This bit reports that whether DMA is in IDLE state or not. 0: DMA is not IDLE 1: DMA is IDLE **Note**: Accessible in both device and host modes.

30	DMABSY	DMA Busy This bit reports that whether DMA is busy. 0: DMA is not busy 1: DMA is busy **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 Indicates which Tx FIFO will be flushed when TXFF bit in the same register is set. Host Mode: 00000: Only non-periodic Tx FIFO is flushed 00001: Only periodic Tx FIFO is flushed 1XXXX: Both periodic and non-periodic Tx FIFOs are flushed Other: Non data FIFO is flushed Device Mode 00000: Only Tx FIFO0 is flushed 00001: Only Tx FIFO1 is flushed … 00101: Only Tx FIFO5 is flushed 1XXXX: All Tx FIFOs are flushed Other: Non data FIFO is flushed
5	TXFF	Tx FIFO flush 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. **Note**: Accessible in both device and host modes.
4	RXFF	Rx FIFO flush 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. **Note**: Accessible in both device and host modes.
3	Reserved	Must be kept at reset value.
2	HFCRST	Host frame counter reset 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. **Note**: Only accessible in host mode.
1	HCSRST	HCLK soft reset Set by the application to reset AHB clock domain circuit. 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	Reserved	PTXFEIF	HCIF	HPIF	Reserved		PXNCIF/ ISOONCIF	ISOINCIF	OEPIF	IEPIF	Reserved	
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 This interrupt is triggered when a resume signal (in device mode) or a remote wakeup signal (in host mode) is detected on the USB. **Note**: Accessible in both device and host modes.
30	SESIF	Session interrupt flag 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). **Note**: Accessible in both device and host modes.
29	DISCIF	Disconnect interrupt flag This interrupt is triggered after a device disconnection. **Note**: Only accessible in host mode.
28	IDPSC	ID pin status change Set by the core when ID status changes. **Note**: Accessible in both device and host modes.
27	Reserved	Must be kept at reset value.
26	PTXFEIF	Periodic Tx FIFO empty interrupt flag 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. **Note**: Only accessible in host mode.
25	HCIF	Host channels interrupt flag 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. **Note**: Only accessible in host mode.
24	HPIF	Host port interrupt flag 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. **Note**: Only accessible in host mode.
23:22	Reserved	Must be kept at reset value.
21	PXNCIF	Periodic transfer Not Complete Interrupt flag 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 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 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) **Note**: Only accessible in device mode.
19	OEPIF	OUT endpoint interrupt flag 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 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. **Note**: Only accessible in device mode.
17:16	Reserved	Must be kept at reset value.
15	EOPFIF	End of periodic frame interrupt flag 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. **Note**: Only accessible in device mode.
14	ISOOPDIF	Isochronous OUT packet dropped interrupt flag 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. **Note**: Only accessible in device mode.
13	ENUMF	Enumeration finished USBHS sets this bit after the speed enumeration finishes. Software is able to read USBHS_DSTAT register to get the current device speed. **Note**: Only accessible in device mode.
12	RST	USB reset USBHS sets this bit when it detects a USB reset signal on bus. **Note**: Only accessible in device mode.
11	SP	USB suspend USBHS sets this bit when it detects that the USB bus is idle for 3 ms and enters suspend state. **Note**: Only accessible in device mode.
10	ESP	Early suspend USBHS sets this bit when it detects that the USB bus is idle for 3 ms. **Note**: Only accessible in device mode.
9:8	Reserved	Must be kept at reset value.
7	GONAK	Global OUT NAK effective 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. **Note**: Only accessible in device mode.
6	GNPINAK	Global IN Non-Periodic NAK effective 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. **Note**: Only accessible in device mode.
5	NPTXFEIF	Non-Periodic Tx FIFO empty interrupt flag 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.

		Note: Only accessible in host mode.
4	RXFNEIF	Rx FIFO non-empty interrupt flag USBHS sets this bit when there is at least one packet or status entry in the Rx FIFO. **Note**: Accessible in both host and device modes.
3	SOF	Start of frame 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. 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. **Note**: Accessible in both host and device modes.
2	OTGIF	OTG interrupt flag 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. **Note**: Accessible in both host and device modes.
1	MFIF	Mode fault interrupt flag 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. **Note**: Accessible in both host and device modes.
0	COPM	Current operation mode 0: Device mode 1: Host mode **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	Reserved	PTXFEIE	HCIE	HPIE	Reserved		PXNCIE/ ISOONCIE	ISOINCIE	OEPIE	IEPIE	Reserved	

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 0: Disable wakeup interrupt 1: Enable wakeup interrupt **Note**: Accessible in both host and device modes.
30	SESIE	Session interrupt enable 0: Disable session interrupt 1: Enable session interrupt **Note**: Accessible in both host and device modes.
29	DISCIE	Disconnect interrupt enable 0: Disable disconnect interrupt 1: Enable disconnect interrupt **Note**: Only accessible in device mode.
28	IDPSCIE	ID pin status change interrupt enable 0: Disable connector ID pin status interrupt 1: Enable connector ID pin status interrupt **Note**: Accessible in both host and device modes.
27	Reserved	Must be kept at reset value.
26	PTXFEIE	Periodic Tx FIFO empty interrupt enable 0: Disable periodic Tx FIFO empty interrupt 1: Enable periodic Tx FIFO empty interrupt **Note**: Only accessible in host mode.
25	HCIE	Host channels interrupt enable 0: Disable host channels interrupt 1: Enable host channels interrupt **Note**: Only accessible in host mode.
24	HPIE	Host port interrupt enable 0: Disable host port interrupt 1: Enable host port interrupt **Note**: Only accessible in host mode.
23:22	Reserved	Must be kept at reset value.
21	PXNCIE	Periodic transfer not complete interrupt enable

		0: Disable Periodic transfer not complete interrupt 1: Enable Periodic transfer not complete interrupt **Note**: Only accessible in host mode.
	ISOONCIE	Isochronous OUT transfer not complete interrupt enable 0: Disable Isochronous OUT transfer not complete interrupt 1: Enable Isochronous OUT transfer not complete interrupt **Note**: Only accessible in device mode.
20	ISOINCIE	Isochronous IN transfer not complete interrupt enable 0: Disable Isochronous IN transfer not complete interrupt 1: Enable Isochronous IN transfer not complete interrupt **Note**: Only accessible in device mode.
19	OEPIE	OUT endpoints interrupt enable 0: Disable OUT endpoints interrupt 1: Enable OUT endpoints interrupt **Note**: Only accessible in device mode.
18	IEPIE	IN endpoints interrupt enable 0: Disable IN endpoints interrupt 1: Enable IN endpoints interrupt **Note**: Only accessible in device mode.
17:16	Reserved	Must be kept at reset value.
15	EOPFIE	End of periodic frame interrupt enable 0: Disable end of periodic frame interrupt 1: Enable end of periodic frame interrupt **Note**: Only accessible in device mode.
14	ISOOPDIE	Isochronous OUT packet dropped interrupt enable 0: Disable isochronous OUT packet dropped interrupt 1: Enable isochronous OUT packet dropped interrupt **Note**: Only accessible in device mode.
13	ENUMFIE	Enumeration finish enable 0: Disable enumeration finish interrupt 1: Enable enumeration finish interrupt **Note**: Only accessible in device mode.
12	RSTIE	USB reset interrupt enable 0: Disable USB reset interrupt 1: Enable USB reset interrupt **Note**: Only accessible in device mode.
11	SPIE	USB suspend interrupt enable 0: Disable USB suspend interrupt 1: Enable USB suspend interrupt

		Note: Only accessible in device mode.
10	ESPIE	Early suspend interrupt enable 0: Disable early suspend interrupt 1: Enable early suspend interrupt **Note**: Only accessible in device mode.
9:8	Reserved	Must be kept at reset value.
7	GONAKIE	Global OUT NAK effective interrupt enable 0: Disable global OUT NAK interrupt 1: Enable global OUT NAK interrupt **Note**: Only accessible in device mode.
6	GNPINAKIE	Global non-periodic IN NAK effective interrupt enable 0: Disable global non-periodic IN NAK effective interrupt 1: Enable global non-periodic IN NAK effective interrupt **Note**: Only accessible in device mode.
5	NPTXFEIE	Non-periodic Tx FIFO empty interrupt enable 0: Disable non-periodic Tx FIFO empty interrupt 1: Enable non-periodic Tx FIFO empty interrupt **Note**: Only accessible in Host mode.
4	RXFNEIE	Receive FIFO non-empty interrupt enable 0: Disable receive FIFO non-empty interrupt 1: Enable receive FIFO non-empty interrupt **Note**: Accessible in both device and host modes.
3	SOFIE	Start of frame interrupt enable 0: Disable start of frame interrupt 1: Enable start of frame interrupt **Note**: Accessible in both device and host modes.
2	OTGIE	OTG interrupt enable 0: Disable OTG interrupt 1: Enable OTG interrupt **Note**: Accessible in both device and host modes.
1	MFIE	Mode fault interrupt enable 0: Disable mode fault interrupt 1: Enable mode fault interrupt **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)

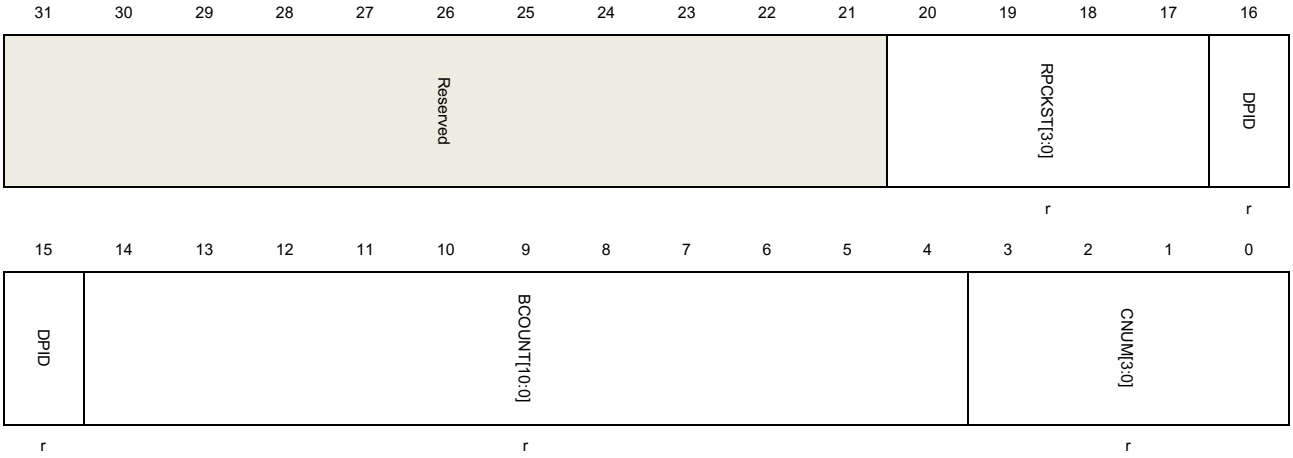
Address offset for Read: 0x001C
Address offset for Pop: 0x0020
Reset value: 0x0000 0000

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)

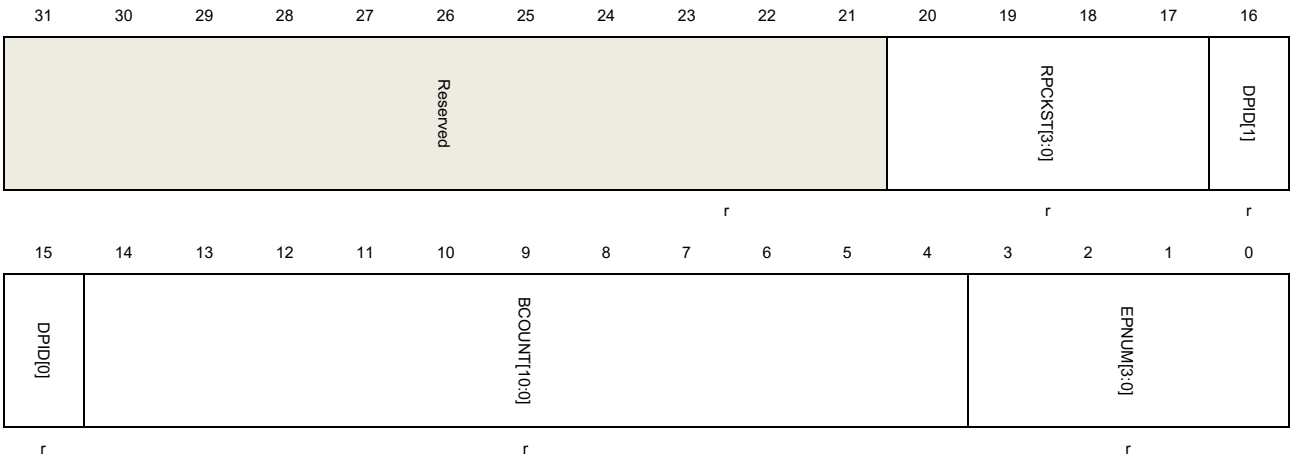
Host mode:



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20:17	RPCKST[3:0]	Received packet status 0010: IN data packet received 0011: IN transfer completed (generates an interrupt if popped) 0101: Data toggle error (generates an interrupt if popped) 0111: Channel halted (generates an interrupt if popped) Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received packet 00: DATA0 10: DATA1 01: DATA2 11: MDATA

14:4	BCOUNT[10:0]	Byte count The byte count of the received IN data packet.
3:0	CNUM[3:0]	Channel number The channel number to which the current received packet belongs.

Device mode:



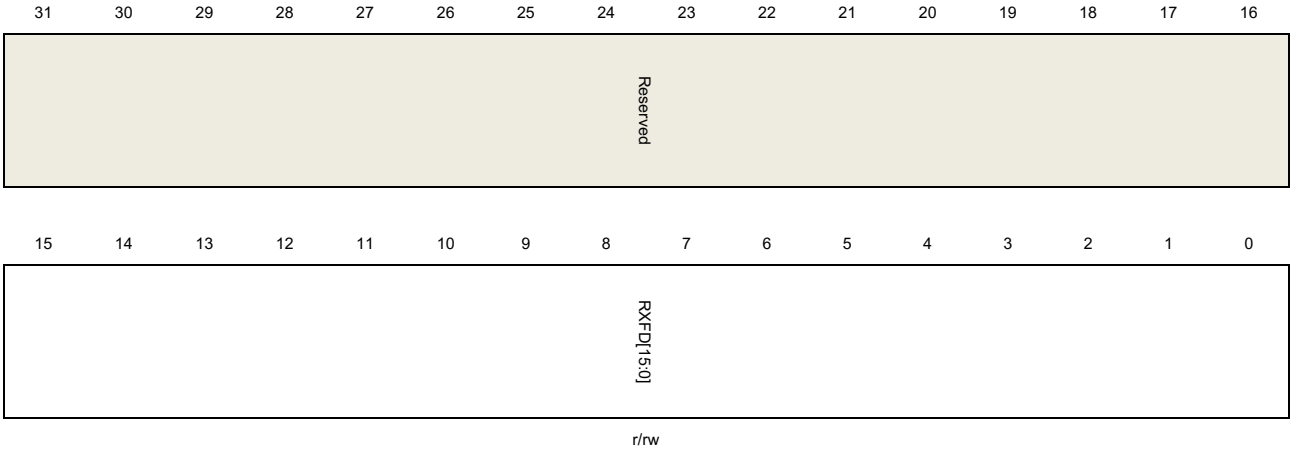
Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20:17	RPCKST[3:0]	Received packet status 0001: Global OUT NAK (generates an interrupt) 0010: OUT data packet received 0011: OUT transfer completed (generates an interrupt) 0100: SETUP transaction completed (generates an interrupt) 0110: SETUP data packet received Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received OUT data packet 00: DATA0 10: DATA1 01: DATA2 11: MDATA
14:4	BCOUNT[10:0]	Byte count 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)



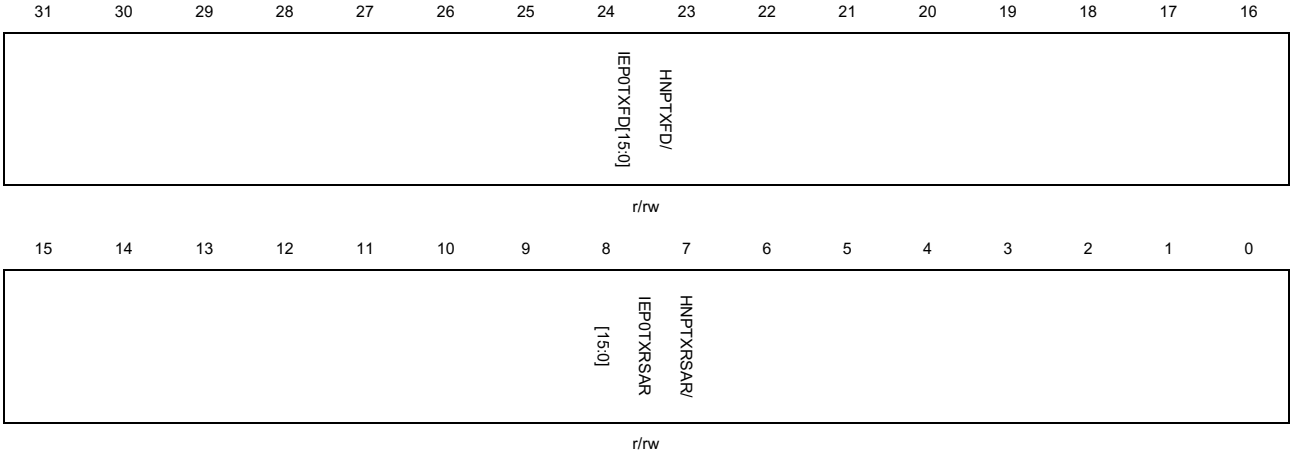
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RXFD[15:0]	Rx FIFO depth In terms of 32-bit word. 1<=RXFD<=1024

Host non-periodic transmit FIFO length register /Device IN endpoint 0 transmit FIFO length (USBHS_HNPTFLEN _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 In terms of 32-bit word. 1<=HNPTXFD<=1024

15:0	HNPTXRSAR[15:0]	Non-periodic Tx RAM start address 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 In terms of 32-bit words. 16<=IEP0TXFD<=140
15:0	IEP0TXRSAR[15:0]	IN Endpoint 0 TX RAM start address The start address for endpoint0 transmit FIFO RAM in terms of 32-bit word.

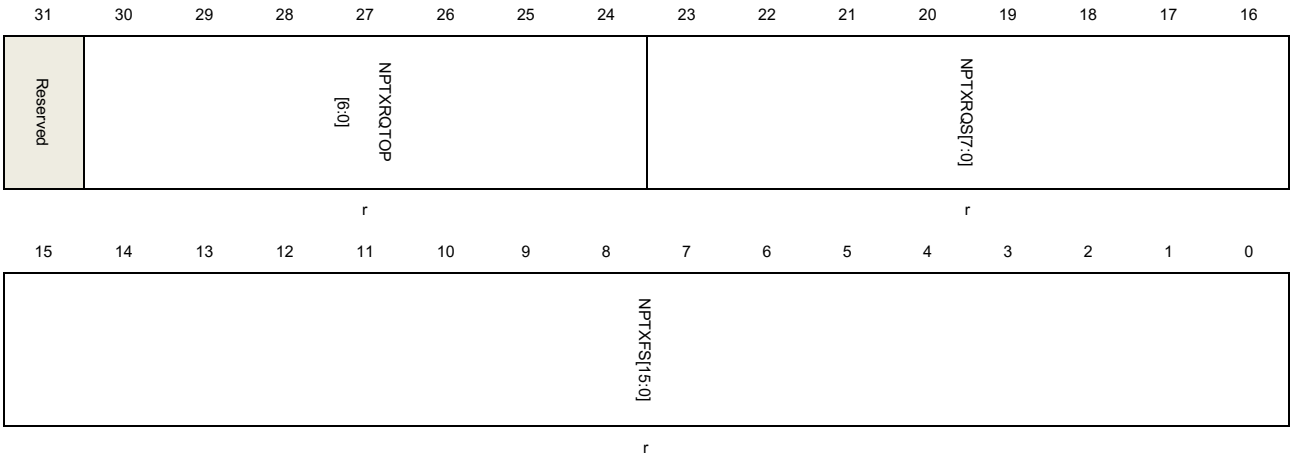
Host non-periodic transmit FIFO/queue status register (USBHS_HNPTFQSTAT)

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 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

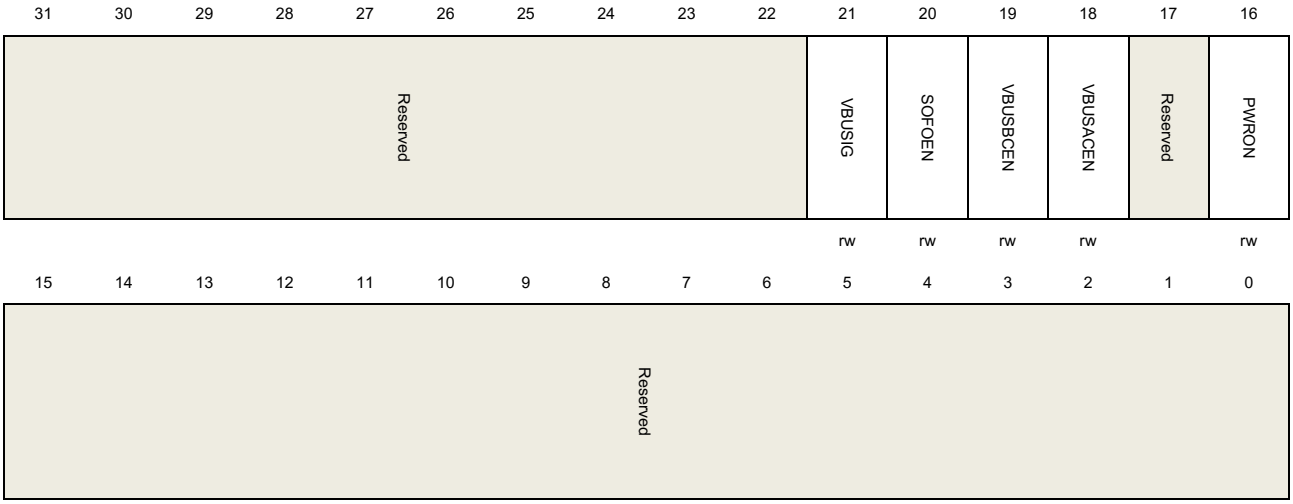
		Bit 24: Terminate Flag, indicating last entry for selected channel.
23:16	NPTXRQS[7:0]	Non-periodic Tx request queue space The remaining space of the non-periodic transmit request queue. 0: Request queue is Full 1: 1 entry 2: 2 entries … n: n entries (0<=n<=8) Others: Reserved
15:0	NPTXFS[15:0]	Non-periodic Tx FIFO space The remaining space of the non-periodic transmit FIFO. In terms of 32-bit words. 0: Non-periodic Tx FIFO is full 1: 1 words 2: 2 words n: n words (0<=n<=NPTXFD) 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	VBUSIG	V_{BUS} Ignored When this bit is set, USBHS doesn't monitor the voltage on VBUS pin and always considers V_{BUS} voltage as valid both in host mode and in device mode, then frees

		the V_{BUS} pin for other usage. 0: VBUS is not ignored. 1: VBUS is ignored and always considers VBUS voltage as valid.
20	SOFOEN	SOF output enable 0: SOF pulse output disabled. 1: SOF pulse output enabled.
19	VBUSBCEN	The V_{BUS} B-device Comparer enable 0: V_{BUS} B-device comparer disabled 1: V_{BUS} B-device comparer enabled
18	VBUSACEN	The VBUS A-device Comparer enable 0: V_{BUS} A-device comparer disabled 1: V_{BUS} A-device comparer enabled
17	Reserved	Must be kept at reset value.
16	PWRON	Power on This bit is the power switch for the internal embedded Full-Speed PHY. 0: Embedded Full-Speed PHY power off. 1: Embedded Full-Speed PHY power on.
15:0	Reserved	Must be kept at reset value.

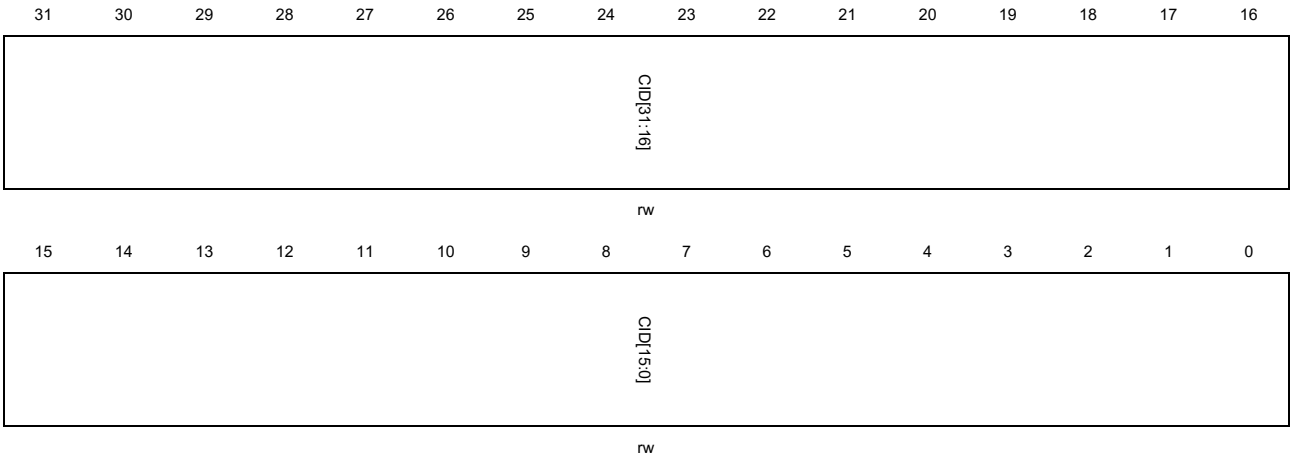
Core ID register (USBHS_CID)

Address offset: 0x003C

Reset value: 0x0000 1000

This register contains the Product ID.

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



Bits	Fields	Descriptions
31:0	CID[31:0]	Core ID Software can write or read this field and uses this field as a unique ID for its

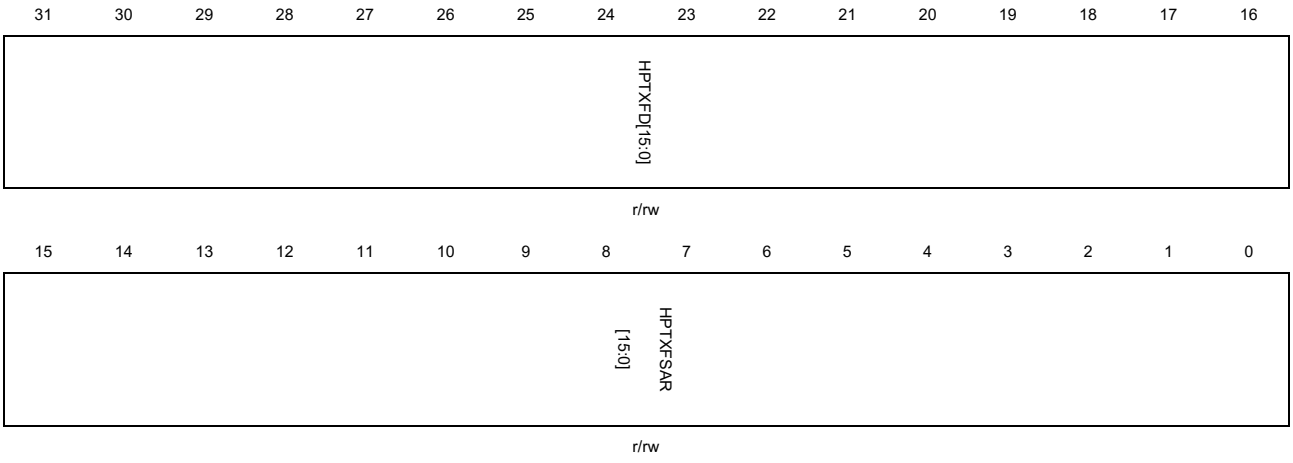
application.

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)



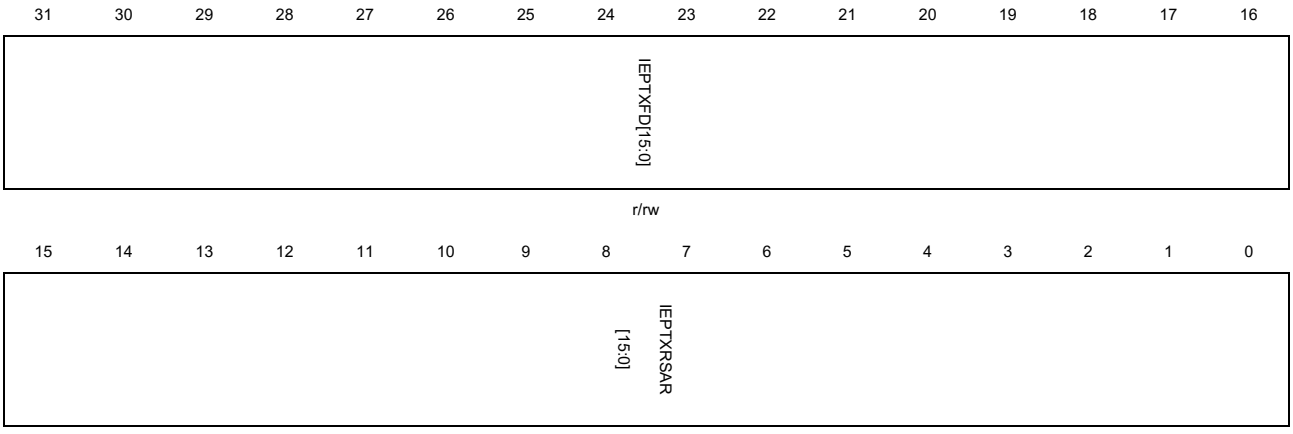
Bits	Fields	Descriptions
31:16	HPTXFD[15:0]	Host Periodic Tx FIFO depth In terms of 32-bit word. 1<=HPTXFD<=1024
15:0	HPTXFSAR[15:0]	Host periodic Tx RAM start address The start address for host periodic transmit FIFO RAM in terms of 32-bit word.

Device IN endpoint transmit FIFO length register (USBHS_DIEPxTFLEN) (x = 1 .. 5, 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)



r/rw

Bits	Fields	Descriptions
31:16	IEPTXFD[15:0]	IN endpoint Tx FIFO x depth In terms of 32-bit word. 1<=IEPTXFD<=1024
15:0	IEPTXRSAR[15:0]	IN endpoint FIFOx Tx x RAM start address The start address for IN endpoint transmit FIFO x in terms of 32-bit word.

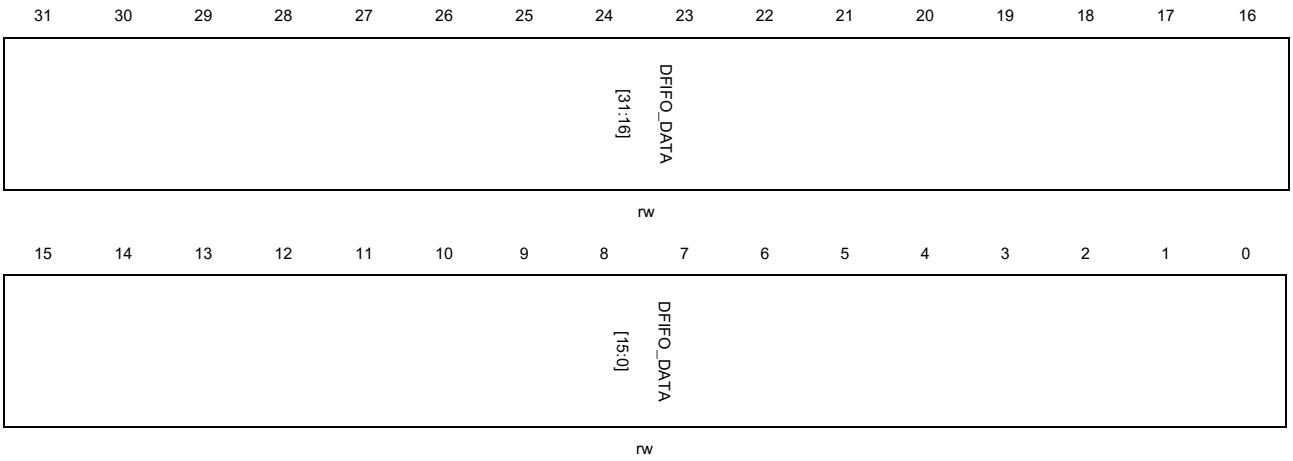
Data FIFO register (USBHS_DFIFO)

Address offset: 0x1000

Host mode: write address range: [0x01000, 0x10FFF], read address range: [0x1000, 0xFFFF]

Device mode: write address range: [0x01000, 0x08FFF], read address range: [0x1000, 0xFFFF]

Reset value: 0x0000 0000



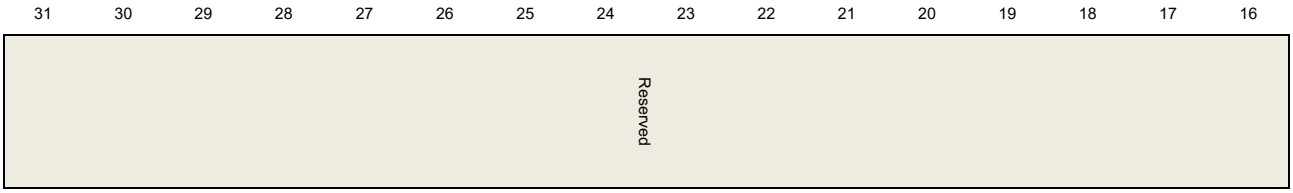
Bits	Fields	Descriptions
31:0	DFIFO_DATA[31:0]	Write the field would push the data into corresponding data FIFO, and read the field would pop the data out corresponding RX FIFO, if read the overflowing address, the last pop data would be gained.

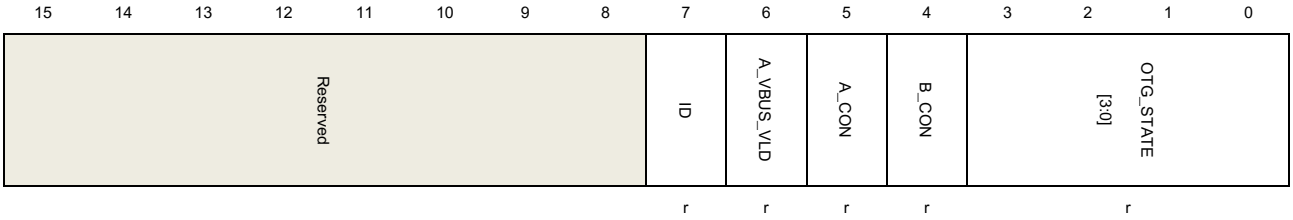
USBFS Debug register (USBHS_DBG)

Address offset: 0x10000

address range: [0x10000, 0x1FFFF]

Reset value: 0x0000 0000





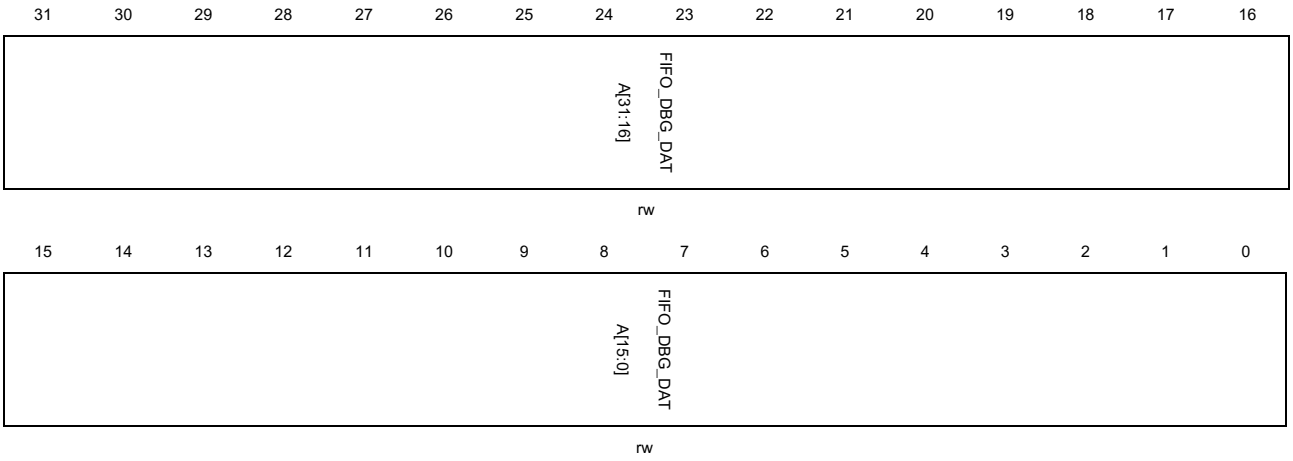
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	ID	ID signal
6	A_VBUS_VLD	Check whether the VBUS voltage exceed valid threshold value
5	A_CON	Detect A device is connected
4	B_CON	Detect B device is connected
3:0	OTG_STATE[3:0]	Internal OTG state of state machine

Data FIFO Debug register (USBHS_DFIFODBG)

Address offset: 0x20000

write address range: [0x20000, 0x20FFF], read address range: [0x20000, 0x2FFFF]

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:0	FIFO_DBG_DATA[31:0]	The register is used to debug FIFO data, not Recommend to users. Write the field would push the data into corresponding data FIFO, and read the field would pop the data out corresponding address, if read the overflowing address range, the last pop data would be gained.

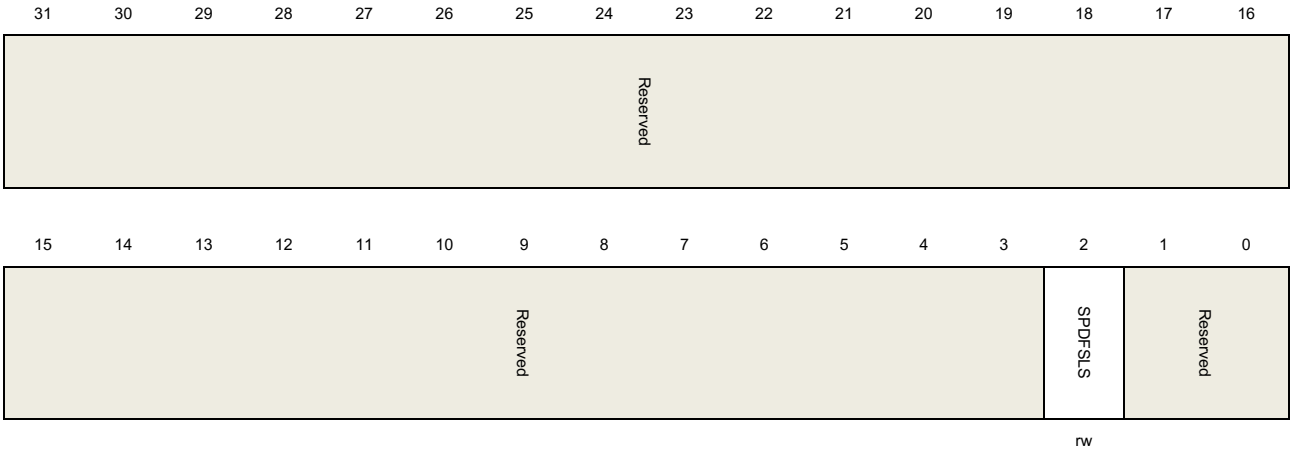
34.7.2. Host control and status registers

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 Software may use this bit to limit USBHS's enumeration speed to FS/LS and make USBHS not perform high-speed enumeration during reset. This bit is only useful with external ULPI PHY, because internal embedded PHY only supports full-speed and low-speed. 0: Speed not limited. 1: Speed limited in FS/LS only.
1:0	Reserved	Must be kept at reset value.

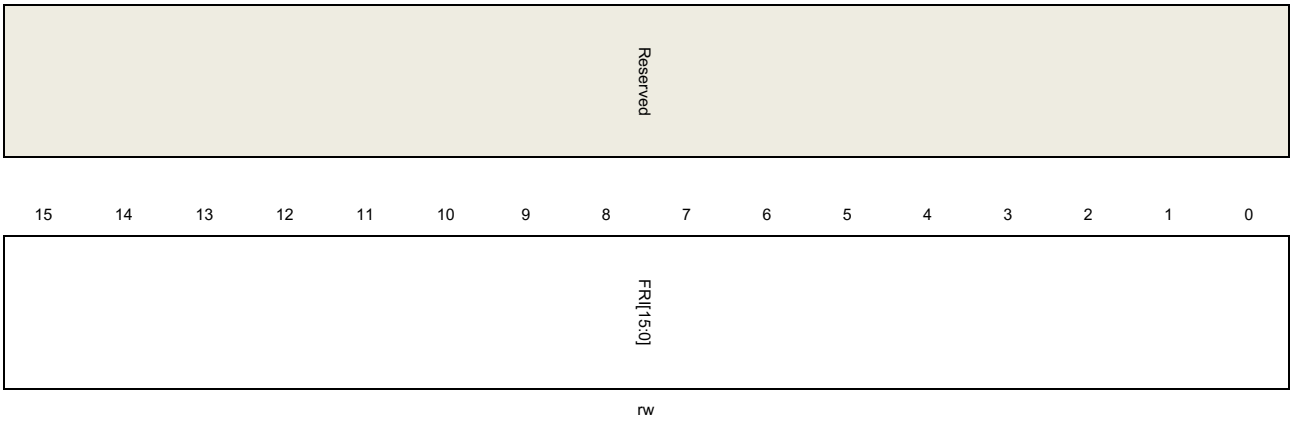
Host frame interval register (USBHS_HFT)

Address offset: 0x0404
Reset value: 0x0000 BB80

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)





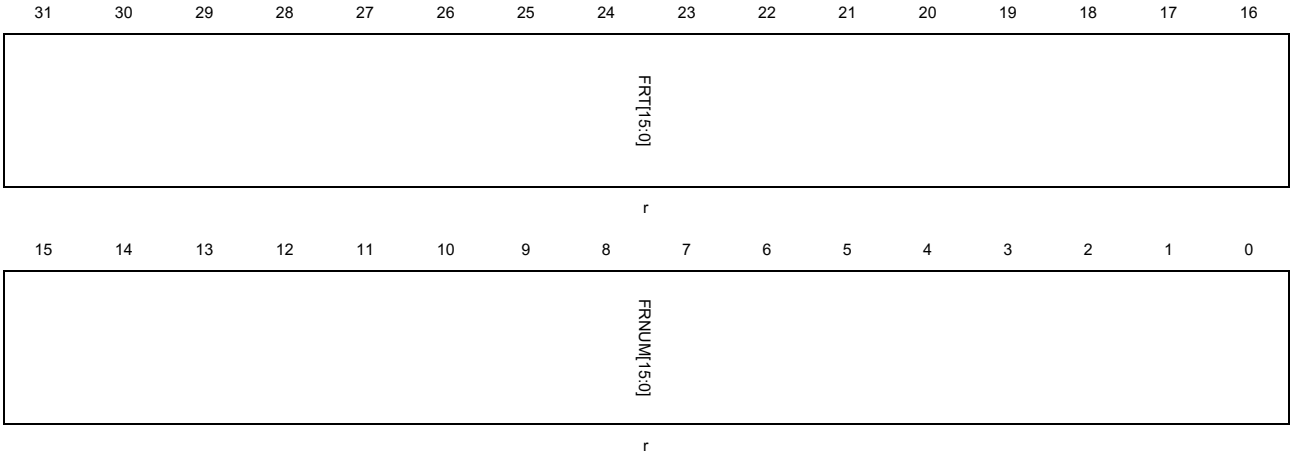
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	FRI[15:0]	Frame interval This value describes the frame time in terms of PHY clock. Port is enabled after a port reset operation, USBHS uses a proper value according to the current speed, and software can write to this field to change the value. This value should be calculated using the frequency described below: Internal Embedded Full-Speed PHY: Full / Low-Speed: 48MHz External ULPI PHY: 60MHz

Host frame information remaining register (USBHS_HFINFR)

Address offset: 0x0408

Reset value: 0xBB80 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.

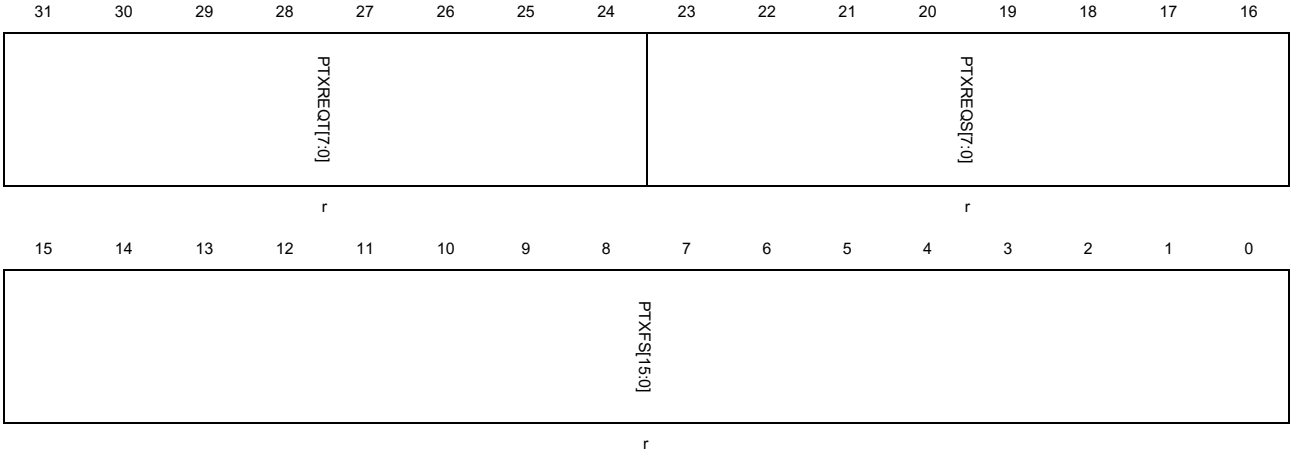
15:0	FRNUM[15:0]	Frame number This field reports the frame number of current frame and returns to 0 after it reaches 0x3FFF.

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 Entry in the 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	PTXREQS[7:0]	Periodic Tx request queue space The remaining space of the periodic transmit request queue. 0: Request queue is Full 1: 1 entry 2: 2 entries … n: n entries (0<=n<=8) Others: Reserved
15:0	PTXFS[15:0]	Periodic Tx FIFO space The remaining space of the periodic transmit FIFO.

In terms of 32-bit word.

0: periodic Tx FIFO is full

1: 1 word

2: 2 words

n: n words (0<=n<=PTXFD)

Others: Reserved

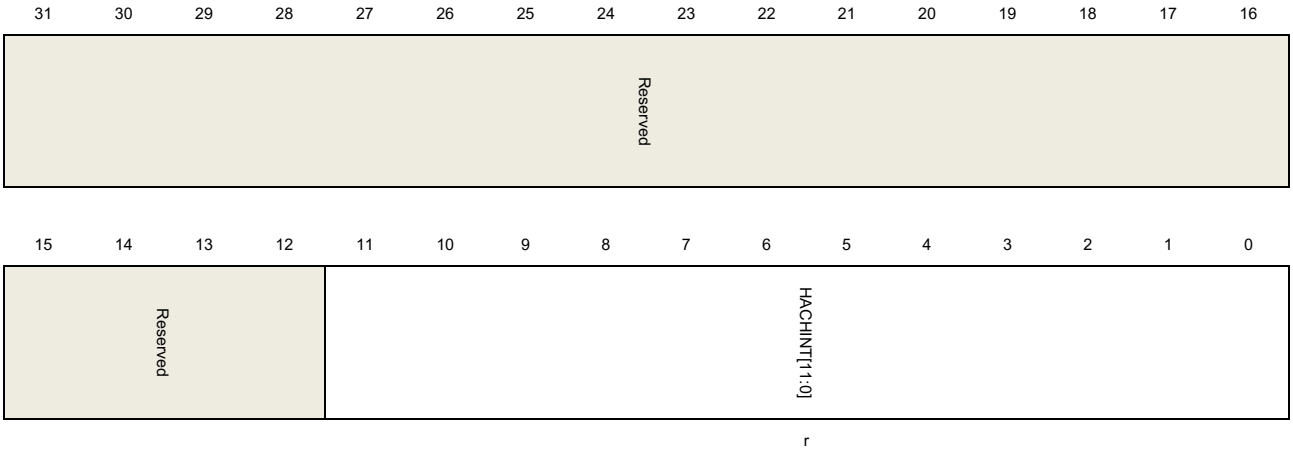
Host all channels interrupt register (USBHS_HACHINT)

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
31:12	Reserved	Must be kept at reset value.
11:0	HACHINT[11:0]	Host all channel interrupts Each bit represents a channel: Bit 0 for channel 0, bit 11 for channel 11.

Host all channels interrupt enable register (USBHS_HACHINTEN)

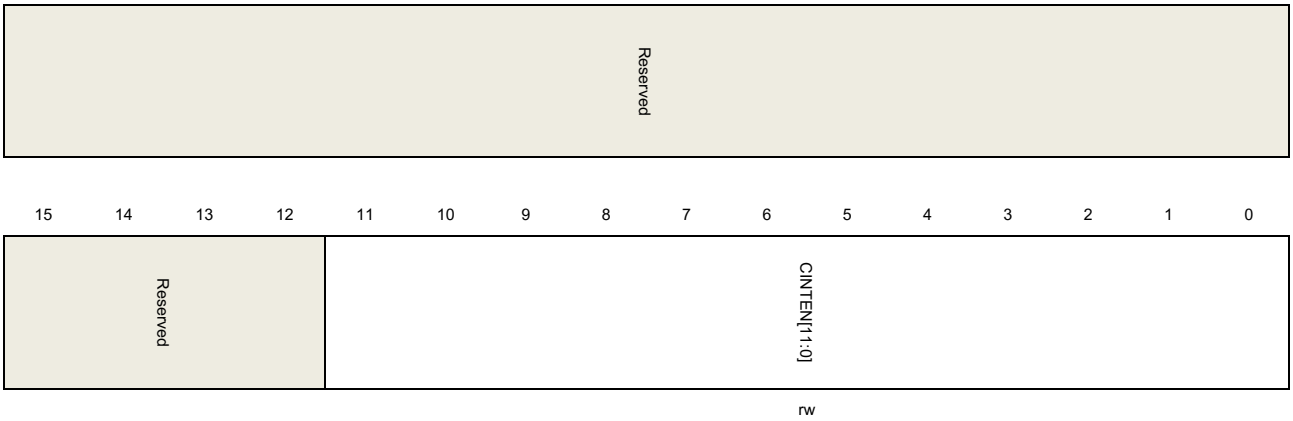
Address offset: 0x0418

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)





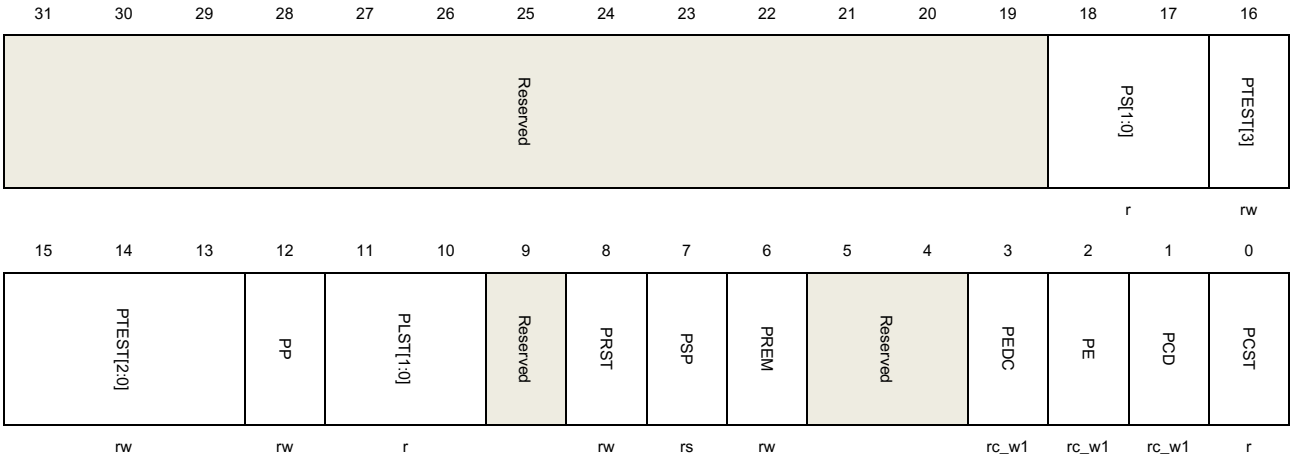
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	CINTEN[11:0]	Channel interrupt enable 0: Disable channel-n interrupt 1: Enable channel-n interrupt Each bit represents a channel: Bit 0 for channel 0, bit 11 for channel 11.

Host port control and status register (USBHS_HPCS)

Address offset: 0x0440
Reset value: 0x0000 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 Report the enumerated speed of the device attached to this port.

		00: High speed 01: Full speed 10: Low speed Others: Reserved
16:13	PTEST[3:0]	Port Test control 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. 0000: Test mode disabled 0001: Test_J mode 0010: Test_K mode 0011: Test_SE0_NAK mode 0100: Test_Packet mode 0101: Test_Force_Enable Others: Reserved
12	PP	Port power 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 VBUS before setting this bit. 0: Port is powered off 1: Port is powered on
11:10	PLST[1:0]	Port line status Report the current state of USB data lines Bit 10: State of DP line Bit 11: State of DM line
9	Reserved	Must be kept at reset value.
8	PRST	Port reset 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. 0: Port is not in reset state 1: Port is in reset state
7	PSP	Port suspend 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: – PRST bit in this register is set by application – PREM bit in this register is set – A remote wakeup signal is detected – A device disconnection is detected 0: Port is not in suspend state 1: Port is in suspend state

6	PREM	Port resume 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. 0: No resume driven 1: Resume driven
5:4	Reserved	Must be kept at reset value.
3	PEDC	Port enable/disable change Set by the core when the status of the Port enable bit 2 in this register changes.
2	PE	Port Enable This bit is automatically set by USBHS after a USB reset signal finishes and cannot be set by software. This bit is cleared by the following events: – A disconnection condition – Software clears this bit 0: Port disabled 1: Port enabled
1	PCD	Port connect detected 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 0: Device is not connected to the port 1: Device is connected to the port

Host channel-x control register (USBHS_HCHxCTL) (x = 0..11, 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
CEN	CDIS	ODDFRM	DAR[6:0]							MPC [1:0]		EPTYPE[1:0]		LSD	Reserved
rs	rs	rw	rw							rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPL[10:0]										
rw	rw				rw										

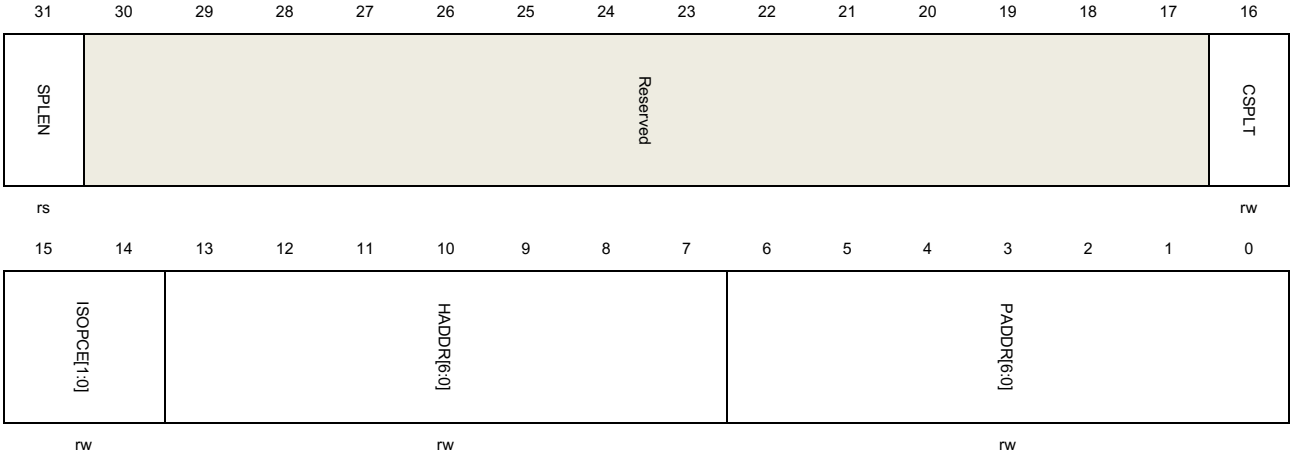
Bits	Fields	Descriptions
31	CEN	Channel enable Set by the application and cleared by USBHS. 0: Channel disabled 1: Channel enabled Software should follow the operation guide to disable or enable a channel.
30	CDIS	Channel disable Software can set this bit to disable the channel from processing transactions. Software should follow the operation guide to disable or enable a channel.
29	ODDFRM	Odd frame 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. 0: Even frame 1: Odd frame
28:22	DAR[6:0]	Device address The address of the USB device that this channel wants to communicate with.
21:20	MPC[1:0]	Multiple Packet Count For periodic transfers, this field indicates to the number of transactions that must be issued per micro-frame. For non-periodic transfers, it defines how many packets the DMA should fetch or write for this channel before the internal DMA engine changes arbitration. 00: Reserved 01: 1 transaction to be issued per micro-frame 10: 2 transactions to be issued per micro-frame 11: 3 transactions to be issued per micro-frame
19:18	EPTYPE[1:0]	Endpoint type The transfer type of the endpoint that this channel wants to communicate with. 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	LSD	Low-Speed device 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 The transfer direction of the endpoint that this channel wants to communicate with. 0: OUT 1: IN

14:11	EPNUM[3:0]	Endpoint number The number of the endpoint that this channel wants to communicate with.
10:0	MPL[10:0]	Maximum packet length The target endpoint's maximum packet length.

Host channel-x split transaction control register (USBHS_HCHxSTCTL) (x = 0..11, 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 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 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 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 00: High-Speed data is the middle of the Full-Speed data payload 01: High-Speed data is the end of the Full-Speed data payload 10: High-Speed data is the beginning of the Full-Speed data payload 11: High-Speed data is all of the Full-Speed data payload
13:7	HADDR[6:0]	HUB address This field contains the USB device address of the hub supporting the specified Full-

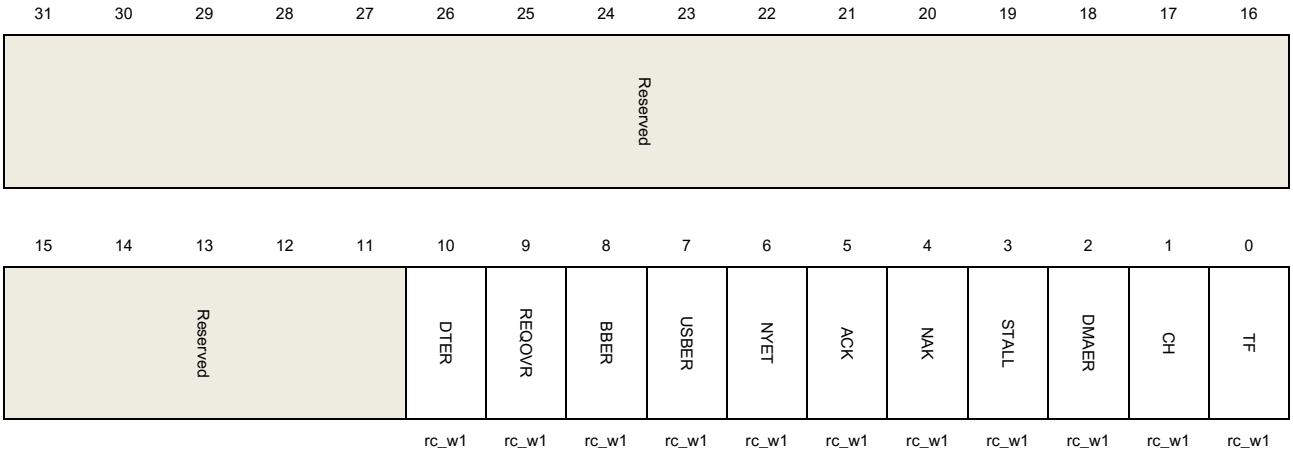
		/Low-Speed device for this Full-/Low-Speed transaction.
6:0	PADDR[6:0]	Port address This field contains the port number of the target hub for this Full-/Low-Speed transaction.

Host channel-x interrupt flag register (USBHS_HCHxINTF) (x = 0 .. 11, 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 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 The periodic request queue is full when software starts new transfer.
8	BBER	Babble error 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 The USB error flag is set when the following conditions occur during receiving a packet:

		– A received packet has a wrong CRC field – A stuff error detected on USB bus – Timeout when waiting for a response packet
6	NYET	NYET A NYET response packet received (in High-Speed).
5	ACK	ACK An ACK response is received or transmitted
4	NAK	NAK A NAK response is received.
3	STALL	STALL A STALL response is received.
2	DMAER	DMA Error An error occurs when DMA tries to fetch or write packet data for this channel.
1	CH	Channel halted When DMA is not enabled: This channel is disabled by the software request. When DMA is enabled: This channel is disabled by DMA because all the transactions of this channel finish successfully or an USB error occurs.
0	TF	Transfer finished 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.

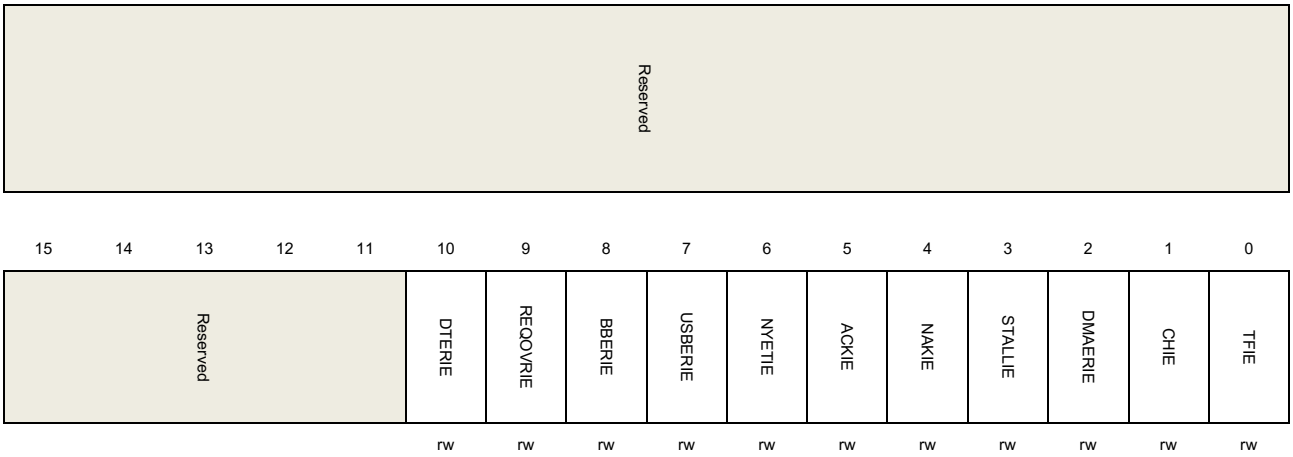
Host channel-x interrupt enable register (USBHS_HCHxINTEN) (x = 0 .. 11, where x = channel number)

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)

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



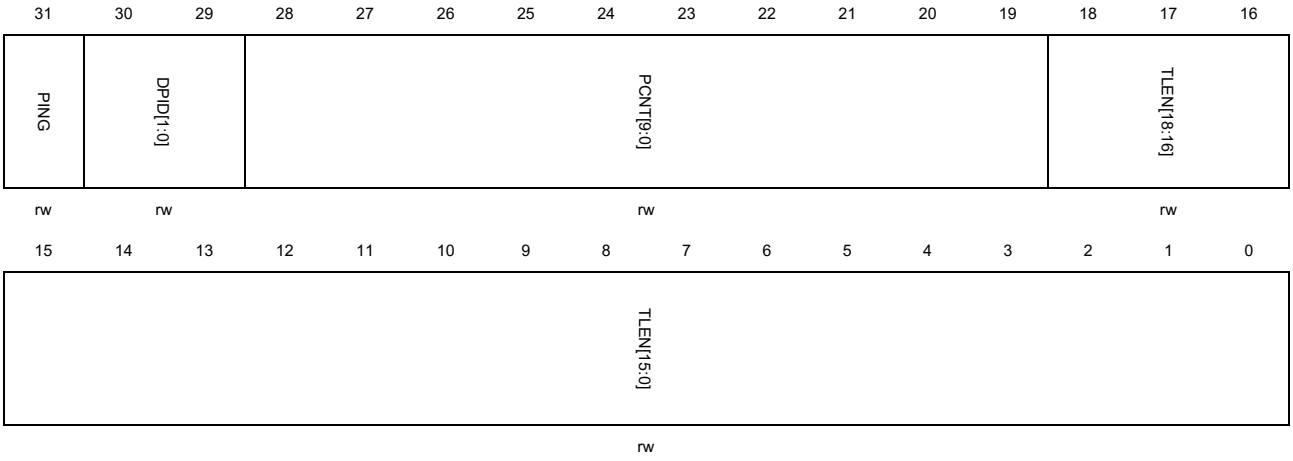
Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10	DTERIE	Data toggle error interrupt enable 0: Disable data toggle error interrupt 1: Enable data toggle error interrupt
9	REQOVRIE	Request queue overrun interrupt enable 0: Disable request queue overrun interrupt 1: Enable request queue overrun interrupt
8	BBERIE	Babble error interrupt enable 0: Disable babble error interrupt 1: Enable babble error interrupt
7	USBERIE	USB bus error interrupt enable 0: Disable USB bus error interrupt 1: Enable USB bus error interrupt
6	NYETIE	NYET interrupt enable 0: Disable NYET interrupt 1: Enable NYET interrupt
5	ACKIE	ACK interrupt enable 0: Disable ACK interrupt 1: Enable ACK interrupt
4	NAKIE	NAK interrupt enable 0: Disable NAK interrupt 1: Enable NAK interrupt
3	STALLIE	STALL interrupt enable 0: Disable STALL interrupt 1: Enable STALL interrupt
2	DMAERIE	DMA Error interrupt enable 0: Disable DMA Error interrupt

		1: Enable DMA Error interrupt
1	CHIE	Channel halted interrupt enable 0: Disable channel halted interrupt 1: Enable channel halted interrupt
0	TFIE	Transfer finished interrupt enable 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

Host channel-x transfer length register (USBHS_HCHxLEN) (x = 0..11, where x = channel number)

Address offset: 0x0510 + (channel_number × 0x20)
Reset value: 0x0000 0000

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



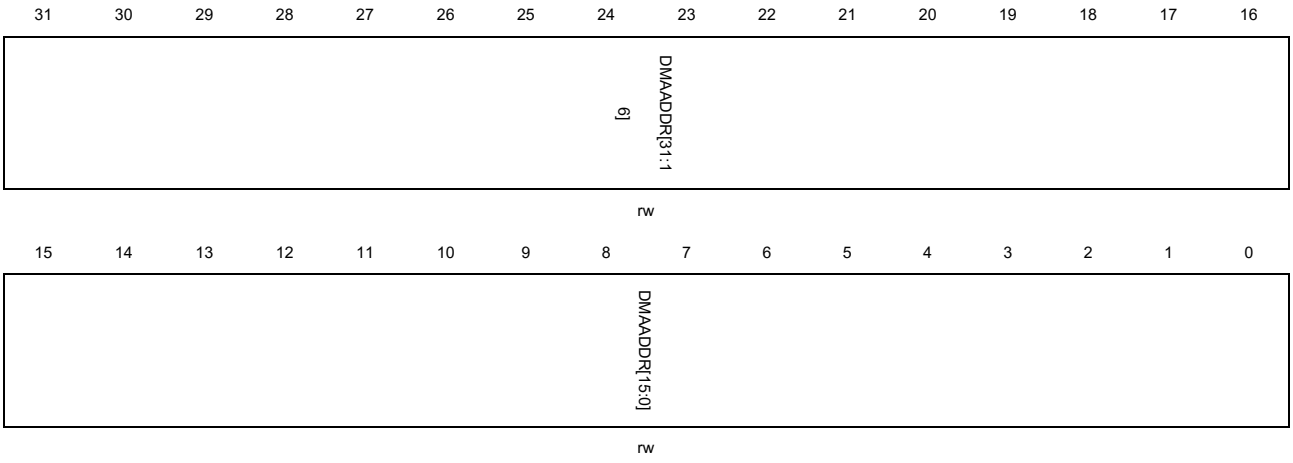
Bits	Fields	Descriptions
31	PING	PING token request For OUT transfer, USBHS will perform PING protocol if software sets this bit. USBHS will automatically set this bit when an OUT transaction receives a NAK or NYET handshake. Do not set this bit for IN transfer.
30:29	DPID[1:0]	Data PID Software should 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 byte 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..11, 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 This field defines the endpoint's DMA address. DMA uses this address to fetch or write packet data for this channel.

34.7.3. Device control and status registers

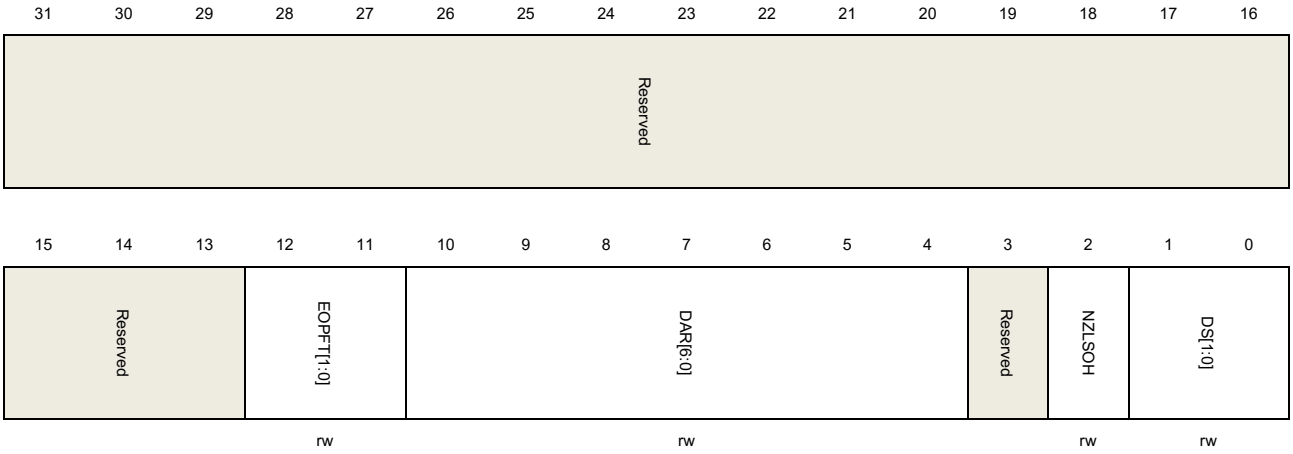
Device configuration register (USBHS_DCFG)

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 This field defines the percentage time point in a frame when the end of periodic frame (EOPF) flag should be triggered. 00: 80% of the frame time 01: 85% of the frame time 10: 90% of the frame time 11: 95% of the frame time
10:4	DAR[6:0]	Device address 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 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.

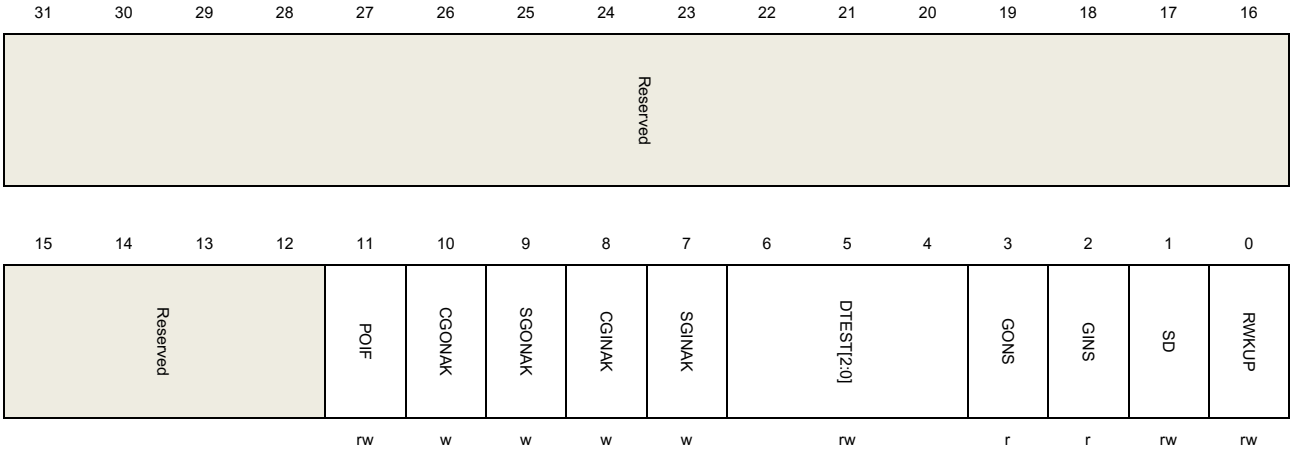
		1: Send a STALL handshake and don't save the received OUT packet.
1:0	DS[1:0]	Device speed This field controls the device speed when the device is connected to a host. 00: High Speed (in external ULPI PHY mode) 01: Full speed Others: Reserved

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
31:12	Reserved	Must be kept at reset value.
11	POIF	Power-on initialization finished 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 Software sets this bit to clear GONS bit in this register.
9	SGONAK	Set global OUT NAK Software sets this bit to set GONS bit in this register. 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 Software sets this bit to clear GINS bit in this register.
7	SGINAK	Set global IN NAK Software sets this bit to set GINS bit in this register. When GINS bit is zero, setting this bit will also cause GINAK flag in

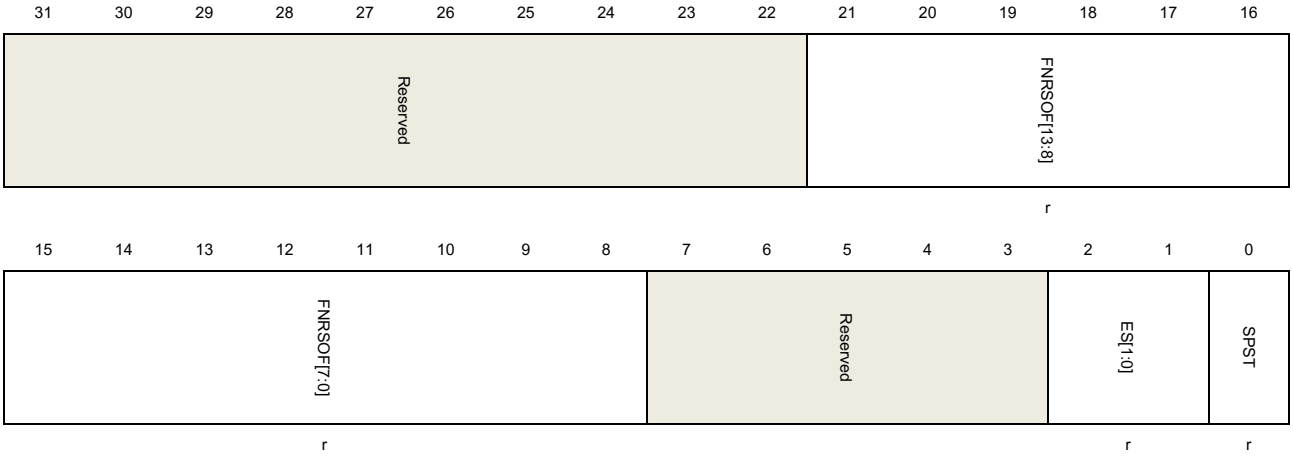
		USBHS_GINTF register triggered after a while. Software should clear the GINAK flag before writing this bit again.
6:4	DTEST[2:0]	Device Test control The software writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is sended on the port. When test mode is used, the HS_CUR_FE bit in USBHS_GUSBCS register should also be set. 000: Test mode disabled 001: Test_J mode 010: Test_K mode 011: Test_SE0_NAK mode 100: Test_Packet mode 101: Test_Force_Enable Others: Reserved
3	GONS	Global OUT NAK status 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. 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 0: The response to IN transaction is decided by Tx FIFO status, endpoint's NAK and STALL bits. 1: USHBS always responses to IN transaction with a NAK handshake.
1	SD	Soft disconnect 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. 0: No soft disconnect generated. 1: Generate a soft disconnect.
0	RWKUP	Remote wakeup 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. 0: No remote wakeup signal generated. 1: Generate remote wakeup signal.

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. 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 This field reports the enumerated device speed. Software should read this field after the ENUMF flag in USBHS_GINTF register is triggered. 00: High speed 01: Full speed Others: reserved
0	SPST	Suspend status This bit reports whether device is in suspend state. 0: Device is not in suspend state. 1: Device is in suspend state.

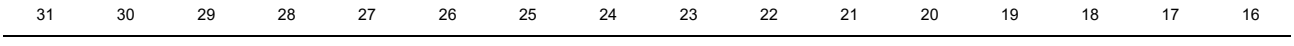
Device IN endpoint common interrupt enable register (USBHS_DIEPINTEN)

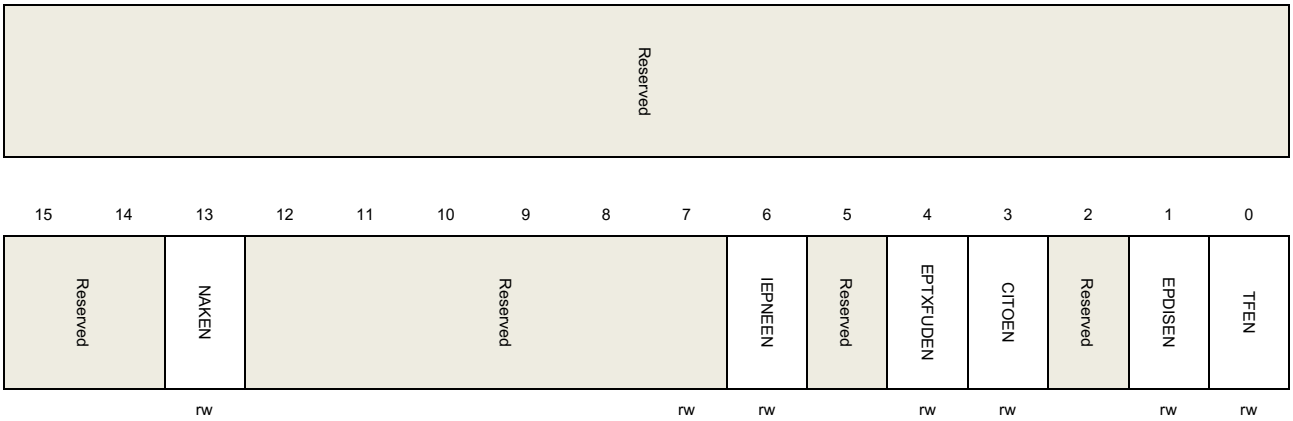
Address offset: 0x810

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 0: Disable interrupt 1: Enable interrupt
12:7	Reserved	Must be kept at reset value.
6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	CITOEN	Control IN Timeout interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

Device OUT endpoint common interrupt enable register (USBHS_DOEPINTEN)

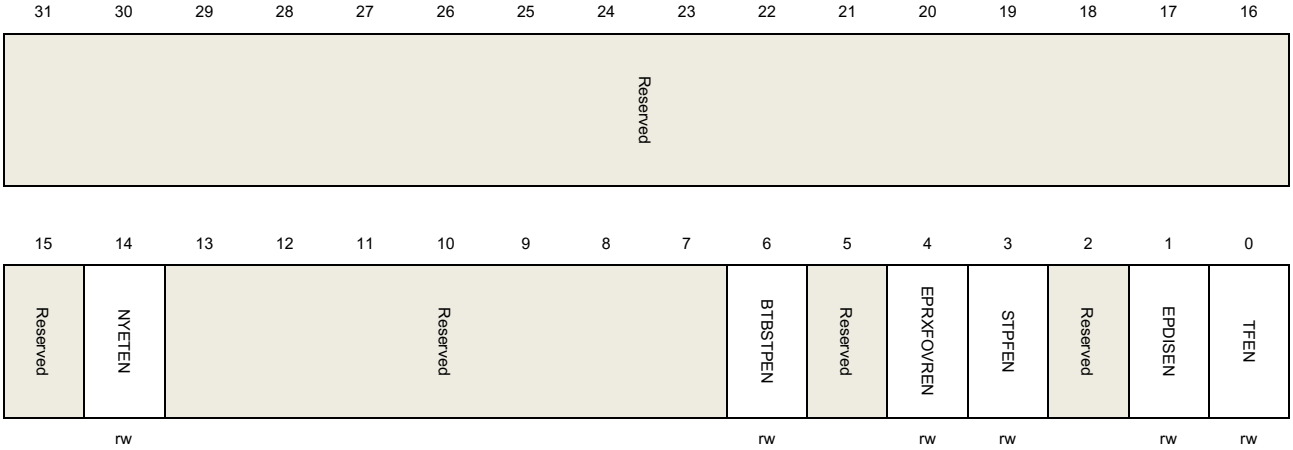
Address offset: 0x0814

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 0: Disable interrupt 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 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPRXFOVREN	Endpoint Rx FIFO overrun interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt

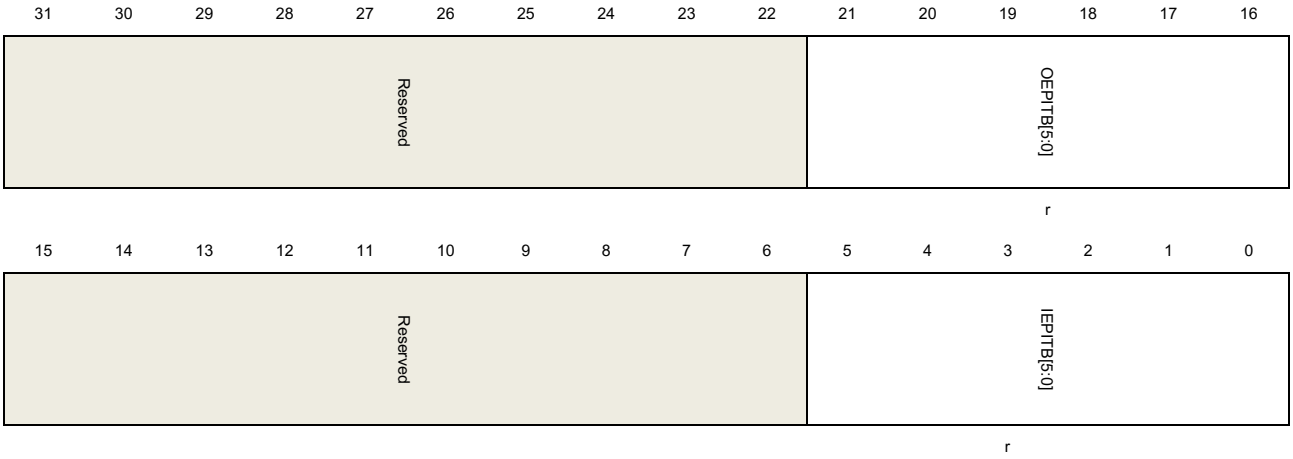
1: Enable interrupt

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
31:22	Reserved	Must be kept at reset value.
21:16	OEPITB[5:0]	Device all OUT endpoints interrupt bits Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 21 for OUT endpoint 5.
15:6	Reserved	Must be kept at reset value.
5:0	IEPITB[5:0]	Device all IN endpoints interrupt bits Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 5 for IN endpoint 5.

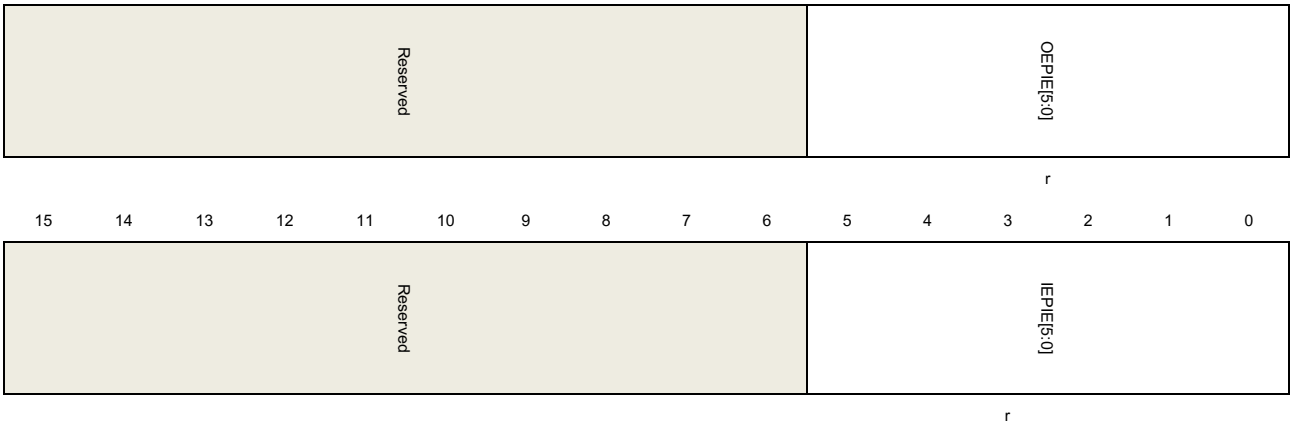
Device all endpoints interrupt enable register (USBHS_DAEPINTEN)

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)

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



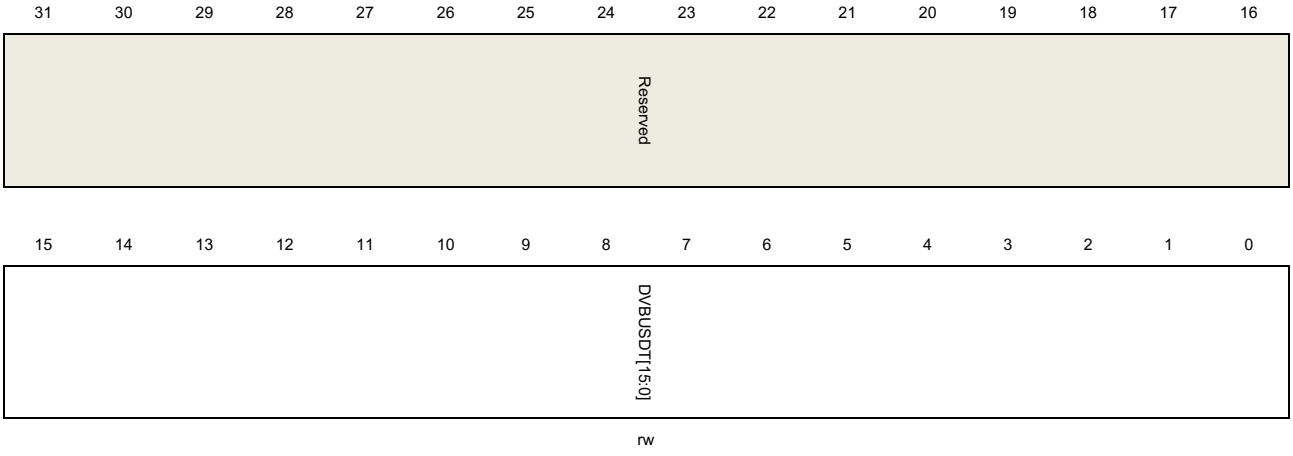
Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value.
21:16	OEPIE[5:0]	Out endpoint interrupt enable 0: Disable OUT endpoint-n interrupt 1: Enable OUT endpoint-n interrupt Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 21 for OUT endpoint 5.
15:6	Reserved	Must be kept at reset value.
5:0	IEPIE[5:0]	IN endpoint interrupt enable bits 0: Disable IN endpoint-n interrupt 1: Enable IN endpoint-n interrupt Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 5 for IN endpoint 5.

Device VBUS discharge time register (USBHS_DVBUSDT)

Address offset: 0x0828

Reset value: 0x0000 17D7

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



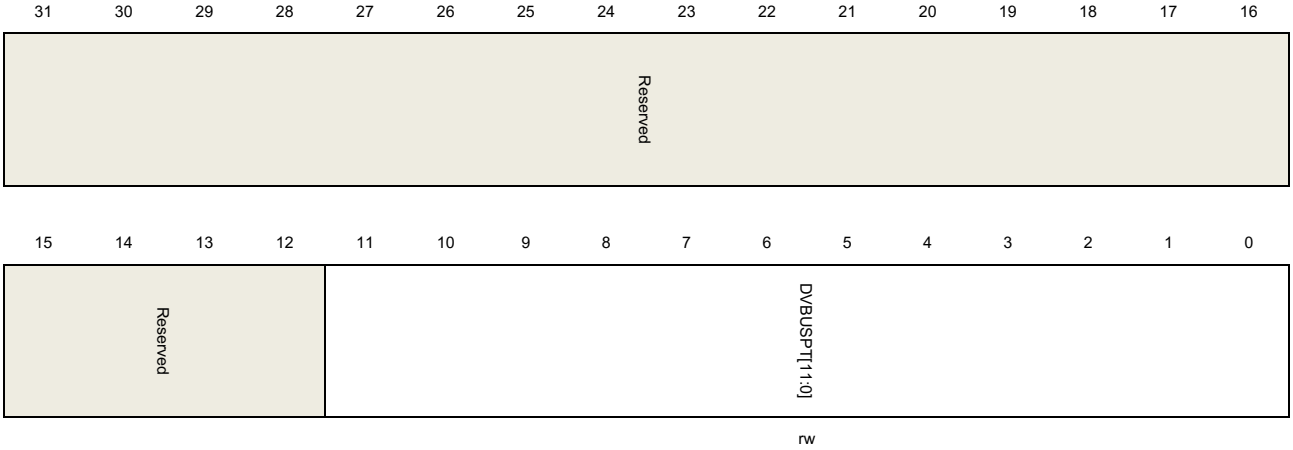
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	DVBUSDT[15:0]	Device V_{BUS} discharge time There is a discharge process after V_{BUS} pulsing in SRP protocol. This field defines the discharge time of V_{BUS}. The true discharge time is 1024*DVBUSDT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ 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 This field defines the pulsing time for V_{BUS}. The true pulsing time is 1024*DVBUSPT[15:0] *$T_{USBCLOCK}$, where $T_{USBCLOCK}$ is the period time of USB clock.

Device threshold control register (USBHS_DTHC)

Address offset: 0x0830

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	APEN	Arbitrator parking enable
26	Reserved	Must be kept at reset value.
25:17	RXTVLEN[8:0]	Receive threshold value length
16	RXTVEN	Receive threshold value enable
15:11	Reserved	Must be kept at reset value.
10:2	TXTVLEN[8:0]	Transmit threshold value length
1	ISOINTVEN	Isochronous IN endpoint threshold value enable
0	NISOINTVEN	Non-isochronous IN endpoint threshold value enable.

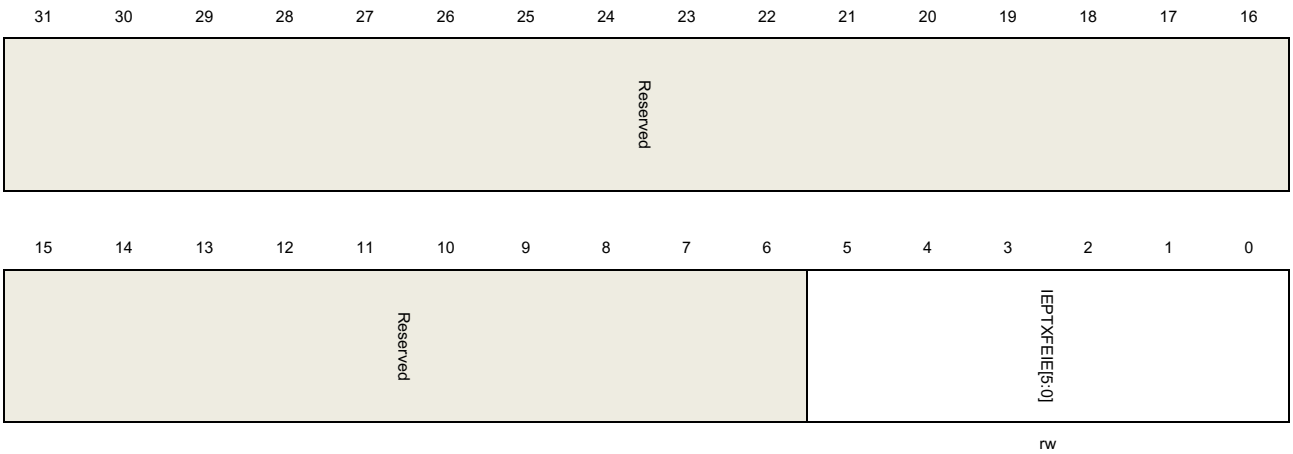
Device IN endpoint FIFO empty interrupt enable register (USBHS_DIEPFEINTEN)

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
31:6	Reserved	Must be kept at reset value.

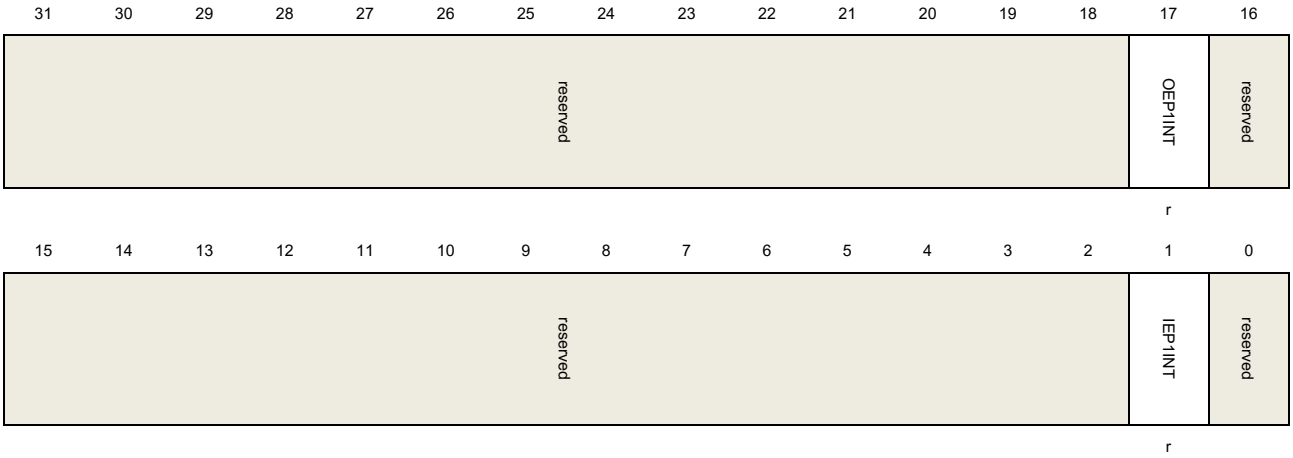
5:0	IEPTXFEIE[5:0]	IN endpoint Tx FIFO empty interrupt enable bits This field controls whether the TXFE bit in USBHS_DIEPxINTF register is able to generate an endpoint interrupt bit in USBHS_DAEPINT register. Bit 0 for IN endpoint 0, bit 5 for IN endpoint 5 0: Disable FIFO empty interrupt 1: Enable FIFO 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)



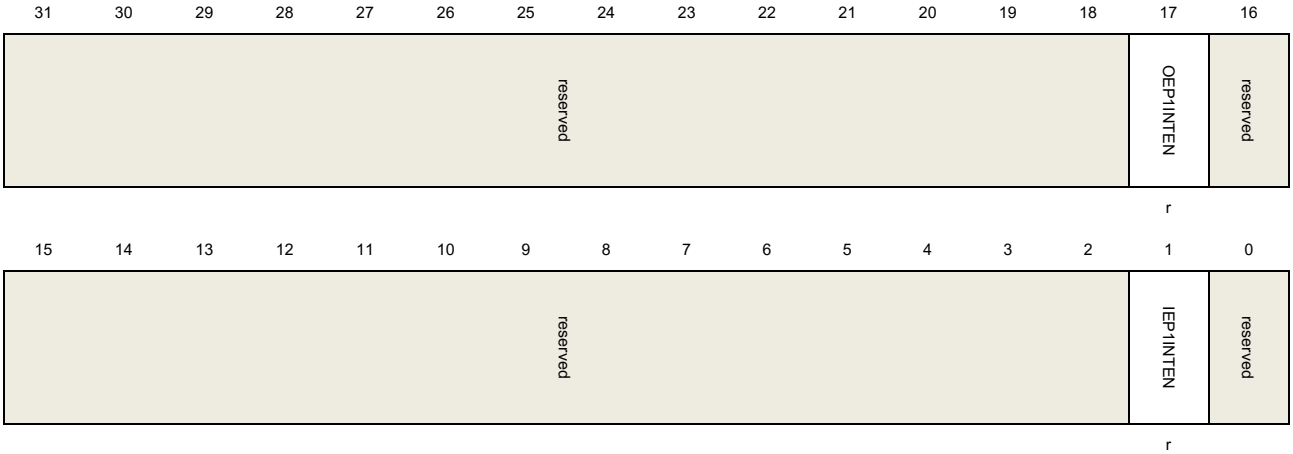
Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value.
17	OEP1INT	OUT Endpoint 1 interrupt
16:2	Reserved	Must be kept at reset value.
1	IEP1INT	IN Endpoint 1 interrupt
0	Reserved	Must be kept at reset value.

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.

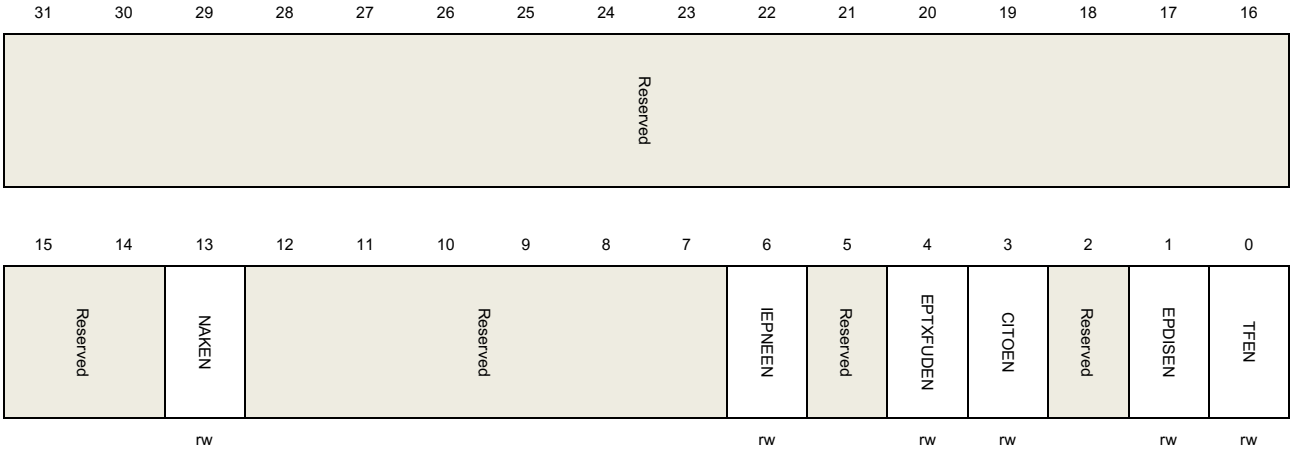
Device IN endpoint-1 interrupt enable register (USBHS_DIEP1INTEN)

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)



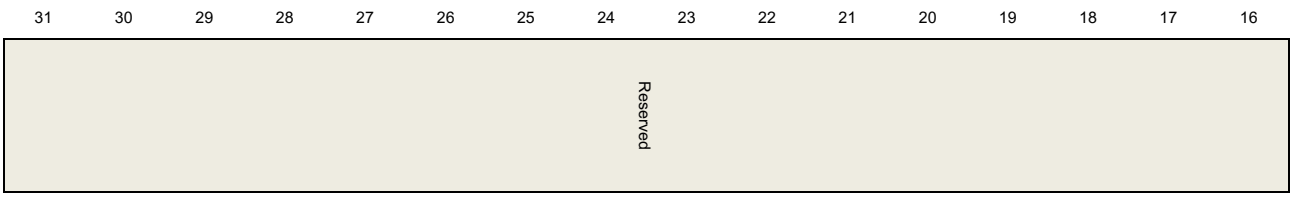
Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13	NAKEN	Interrupt enable bit of NAK handshake sent by USBHS 0: Disable interrupt 1: Enable interrupt
12:7	Reserved	Must be kept at reset value.
6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	CITOEN	Control IN Timeout interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

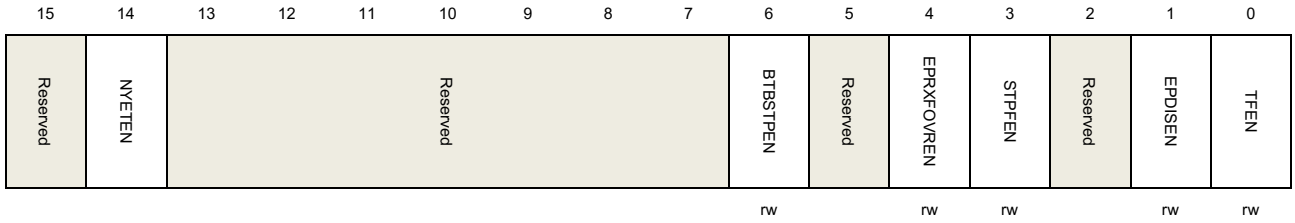
Device OUT endpoint-1 interrupt enable register (USBHS_DOEP1INTEN)

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)





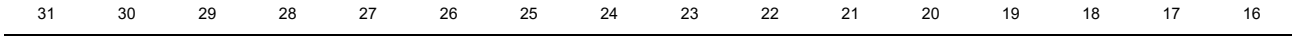
Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	NYETEN	Send NYET handshake interrupt enable bit 0: Disable interrupt 1: Enable interrupt
13:7	Reserved	Must be kept at reset value.
6	BTBSTPEN	Back-to-back SETUP packets (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
5	Reserved	Must be kept at reset value.
4	EPRXFOVREN	Endpoint Rx FIFO over run interrupt enable bit 0: Disable interrupt 1: Enable interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable interrupt 1: Enable interrupt
2	Reserved	Must be kept at reset value.
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable interrupt 1: Enable interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable interrupt 1: Enable interrupt

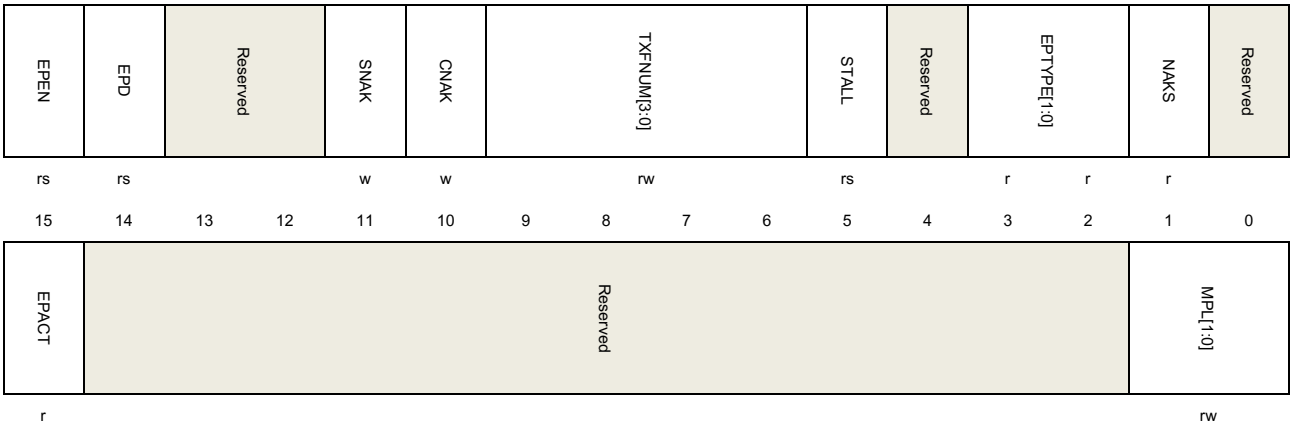
Device IN endpoint 0 control register (USBHS_DIEP0CTL)

Address offset: 0x0900

Reset value: 0x0000 8000

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





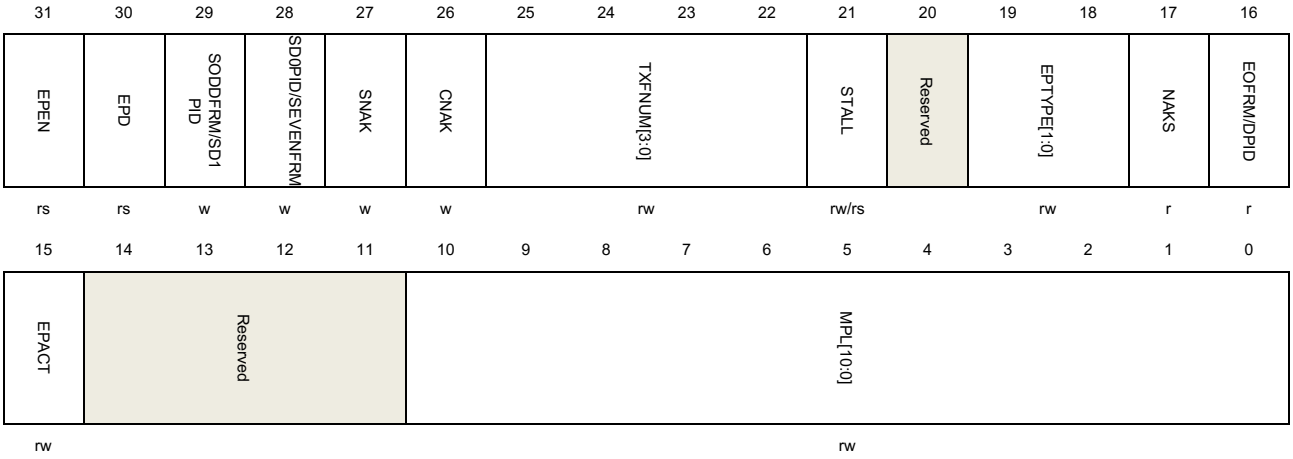
Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBHS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should follow the operation guide to disable or enable an endpoint.
29:28	Reserved	Must be kept at reset value.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Define the Tx FIFO number of IN endpoint 0.
21	STALL	STALL handshake 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 This field is fixed to '00' for control endpoint.
17	NAKS	NAK status This bit controls the NAK status of USBHS when both STALL bit in this register and GINS bit in USBHS_DCTL register are cleared:

		0: USBHS sends data or handshake packets according to the status of the endpoint's Tx FIFO. 1: USBHS always sends NAK handshake to the IN token. 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 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 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: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

Device IN endpoint-x control register (USBHS_DIEPxCTL) (x = 1 .. 5, 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 Set by the application and cleared by USBHS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.

30	EPD	Endpoint disable 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) This bit has effect only if this is an isochronous IN endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk IN endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM	Set even frame (For isochronous IN endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk IN endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Defines the Tx FIFO number of this IN endpoint.
21	STALL	STALL handshake 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. For control IN endpoint: Only USBHS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it. For interrupt or bulk IN endpoint: Only software can clear this bit
20	Reserved	Must be kept at reset value.
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBHS when both STALL bit in this register and GINS bit in USBHS_DCTL register are cleared: 0: USBHS sends data or handshake packets according to the status of the

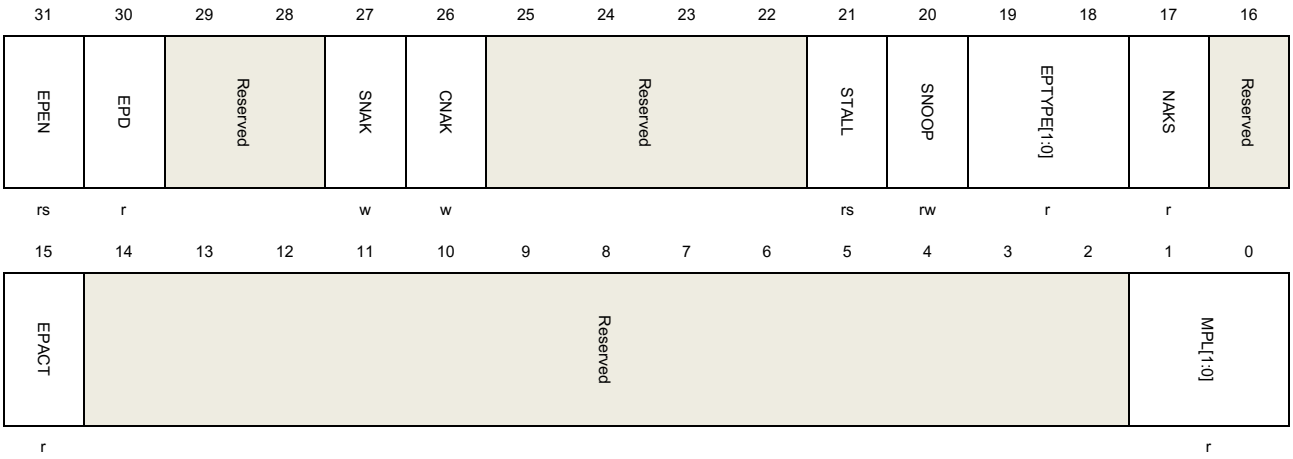
		endpoint's Tx FIFO. 1: USBHS always sends NAK handshake to the IN token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	EOFRM	Even/odd frame (For isochronous IN endpoints) For isochronous transfer, software can use this bit to control that USBHS only 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. 0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk IN endpoints) 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. 0: Data packet's PID is DATA0 1: Data packet's PID is DATA1
15	EPACT	Endpoint active 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 Set by the application and cleared by USBHS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable This bit is fixed to 0 for OUT endpoint 0.
29:28	Reserved	Must be kept at reset value.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register
25:22	Reserved	Must be kept at reset value.
21	STALL	STALL handshake 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 This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBHS doesn't check the received data packet's CRC value. 0: Snoop mode disabled 1: Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type This field is fixed to '00' for control endpoint.
17	NAKS	NAK status This bit controls the NAK status of USBHS when both STALL bit in this register and GONS bit in USBHS_DCTL register are cleared: 0: USBHS sends data or handshake packets according to the status of the endpoint's Rx FIFO. 1: USBHS always sends NAK handshake to the OUT token. 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 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

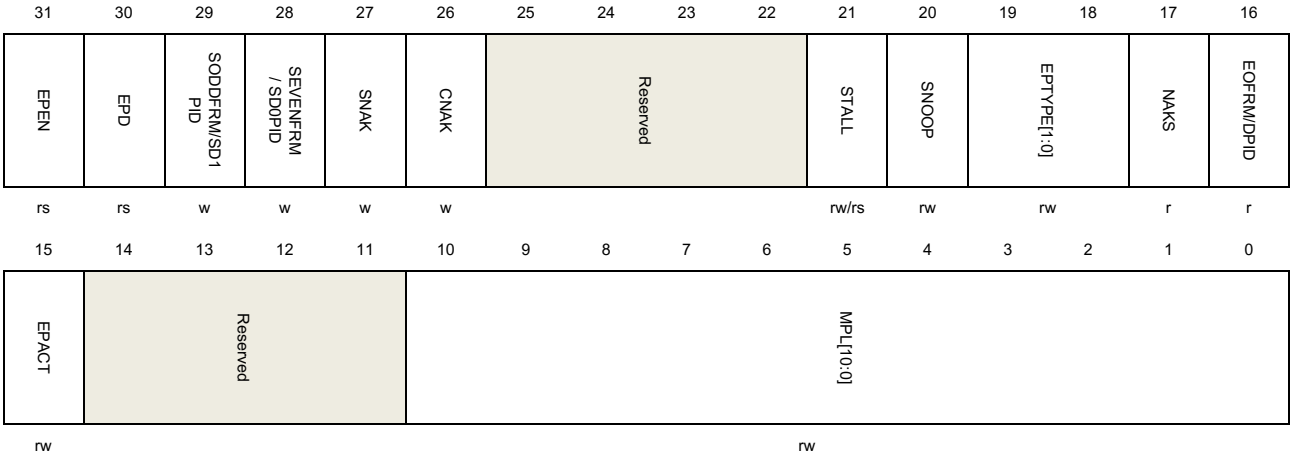
This is a read-only field, and its value comes from the MPL field of USBHS_DIEP0CTL register:
00: 64 bytes
01: 32 bytes
10: 16 bytes
11: 8 bytes

Device OUT endpoint-x control register (USBHS_DOEPxCTL) (x = 1..5, 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)



Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBHS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable 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) This bit has effect only if this is an isochronous OUT endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk OUT endpoints) Software sets this bit to set DPID bit in this register.

28	SEVENFRM	Set even frame (For isochronous OUT endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk OUT endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	Reserved	Must be kept at reset value.
21	STALL	STALL handshake 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. For control OUT endpoint: Only USBHS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it. For interrupt or bulk OUT endpoint: 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. 0: Snoop mode disabled 1: Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBHS when both STALL bit in this register and GONS bit in USBHS_DCTL register are cleared: 0: USBHS sends handshake packets according to the status of the endpoint's Rx FIFO. 1: USBHS always sends NAK handshake to the OUT token. 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) For isochronous transfer, software can use this bit to control that USBHS only

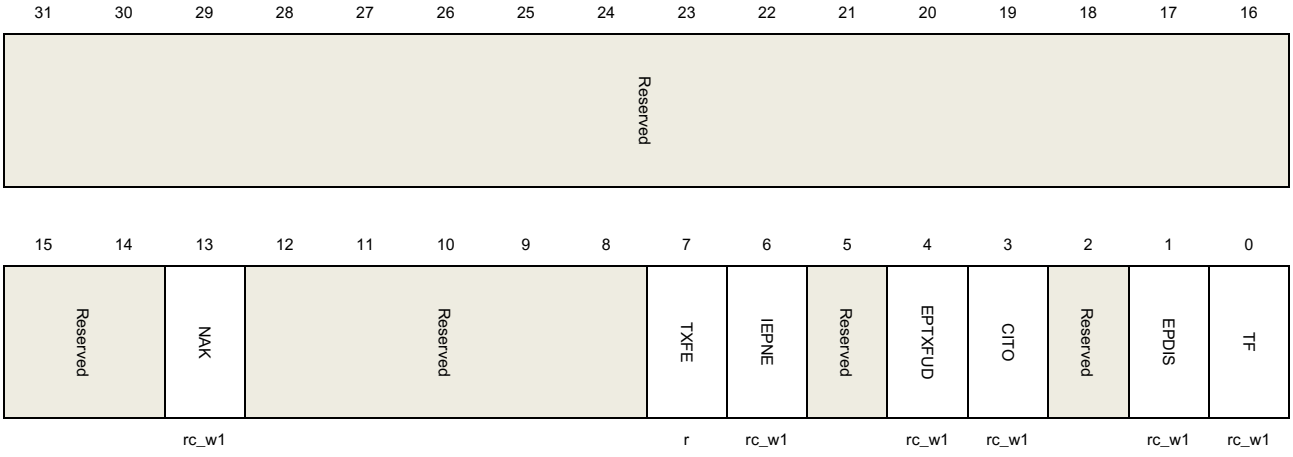
		receives data packets in even or odd frames. If the current frame number's parity doesn't match with this bit, USBHS just drops the data packet. 0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk OUT endpoints) These is a data PID toggle scheme 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. 0: Data packet's PID is DATA0 1: Data packet's PID is DATA1
15	EPACT	Endpoint active 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..5, 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 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 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 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 This flag is triggered if the Tx FIFO has no packet data when an IN token is incoming
3	CITO	Control IN Timeout interrupt 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 This flag is triggered when an endpoint is disabled from the software's request.
0	TF	Transfer finished This flag is triggered when all the IN transactions assigned to this endpoint have finished.

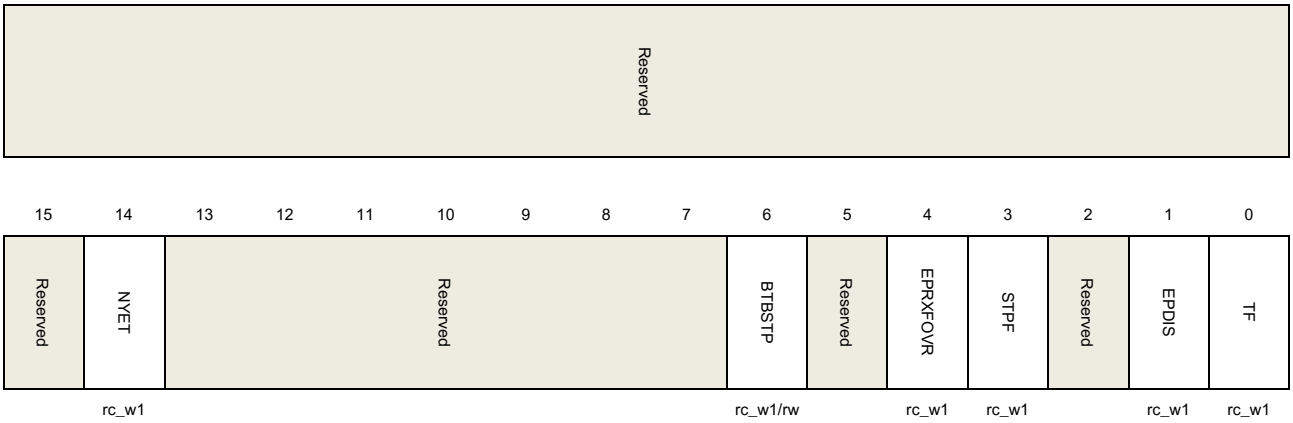
Device OUT endpoint-x interrupt flag register (USBHS_DOEPxINTF) (x = 0 .. 5, where x = endpoint_number)

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)

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



Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	NYET	NYET handshake is sent 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) 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 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 This flag is triggered when an endpoint is disabled from the software's request.
0	TF	Transfer finished This flag is triggered when all the OUT transactions assigned to this endpoint have finished.

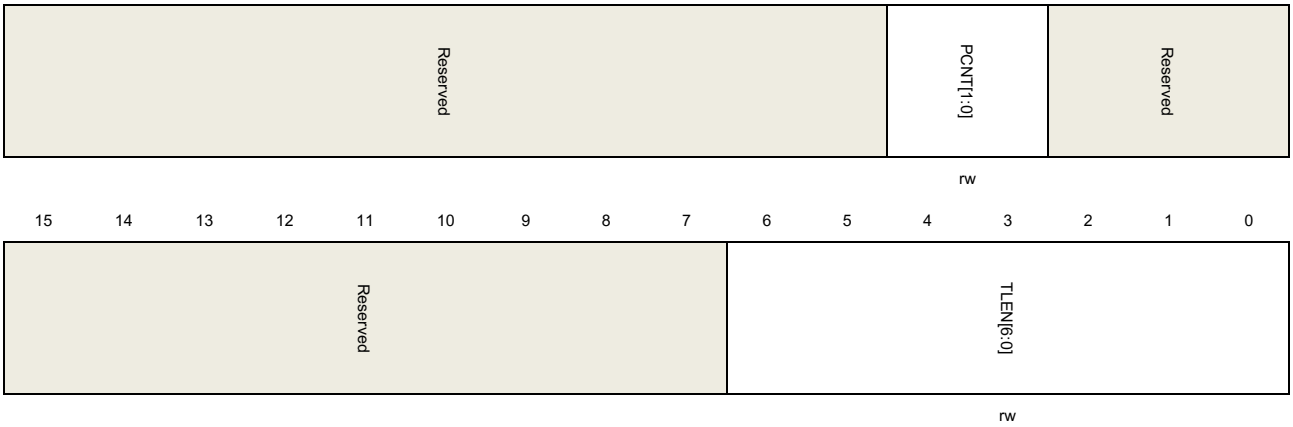
Device IN endpoint 0 transfer length register (USBHS_DIEP0LEN)

Address offset: 0x0910

Reset value: 0x0000 0000

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





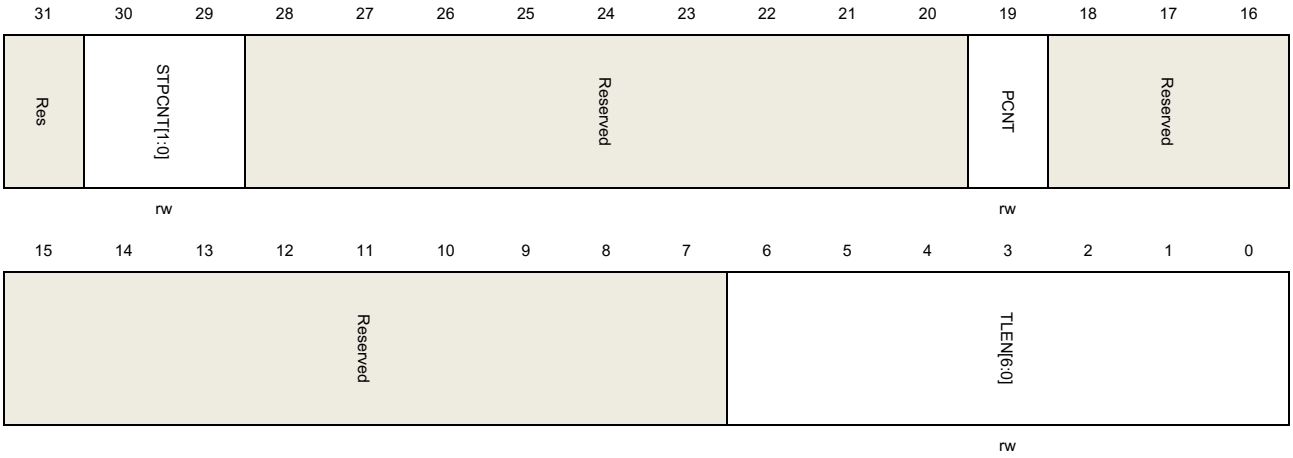
Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value.
20:19	PCNT[1:0]	Packet count The number of data packets desired to be transmitted in a transfer. Software should program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBHS after each successful data packet transmission.
18:7	Reserved	Must be kept at reset value.
6:0	TLEN[6:0]	Transfer length The total data byte number of a transfer. 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)

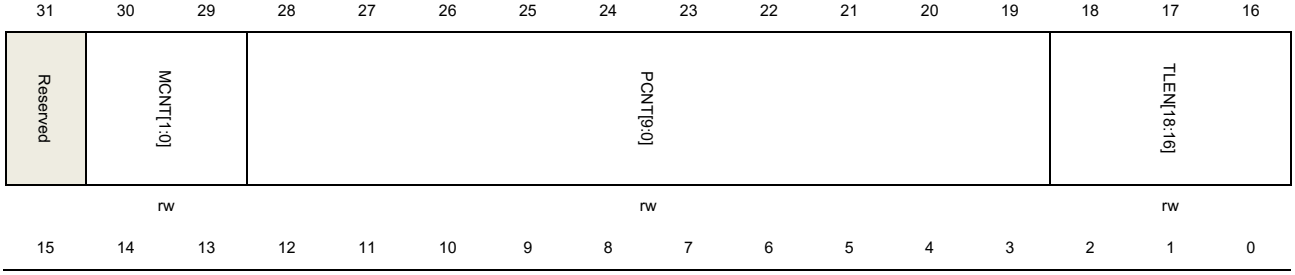


Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	STPCNT[1:0]	SETUP packet count This field defines the maximum number of back-to-back SETUP packets what this endpoint can accept. Software should program this field before setup transfers. Each time a back-to-back setup packet is received, USBHS decreases this field by one. When this field reaches zero, the BTBSTP flag in USBHS_DOEP0INTF register will be triggered. 00: 0 packet 01:1 packet 10: 2 packets 11: 3 packets
28:20	Reserved	Must be kept at reset value.
19	PCNT	Packet count The number of data packets is desired to receive in a transfer. 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 The total data byte 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 Rx FIFO, this field is decreased by the byte size of the packet.

Device IN endpoint-x transfer length register (USBHS_DIEPxLEN) (x = 1 .. 5, where x = endpoint_number)

Address offset: 0x910 + (endpoint_number × 0x20)
Reset value: 0x0000 0000

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



TLEN[15:0]
rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	MCNT[1:0]	Multi count This field indicates the number of packets which should be transmitted in a frame 01: 1 packet 10: 2 packets 11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted in a transfer. 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 The total data byte number of a transfer. 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 .. 5, where x = endpoint_number)

Address offset: 0x0B10 + (endpoint_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
Res	RXDPID/STPCN[1:0]		PCNT[9:0]										TLEN[18:16]		
	r/rw		rw										rw		

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

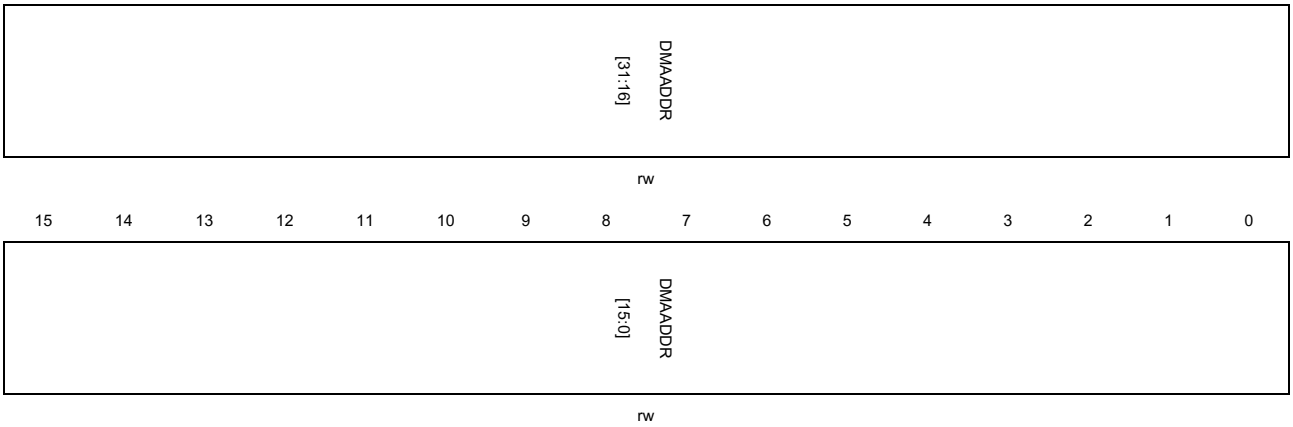
Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30:29	RXDPID[1:0]	Received data PID (For isochronous OUT endpoints) This field saves the PID of the latest received data packet on this endpoint. 00: DATA0 01: DATA2 10: DATA1 11: MDATA
	STPCNT[1:0]	SETUP packet count (For control OUT Endpoints.) This field defines the maximum number back-to-back SETUP packets this endpoint can accept. 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. 00: 0 packet 01: 1 packet 10: 2 packets 11: 3 packets
28:19	PCNT[9:0]	Packet count The number of data packets desired to receive in a transfer. 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 byte 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..5, 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)

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



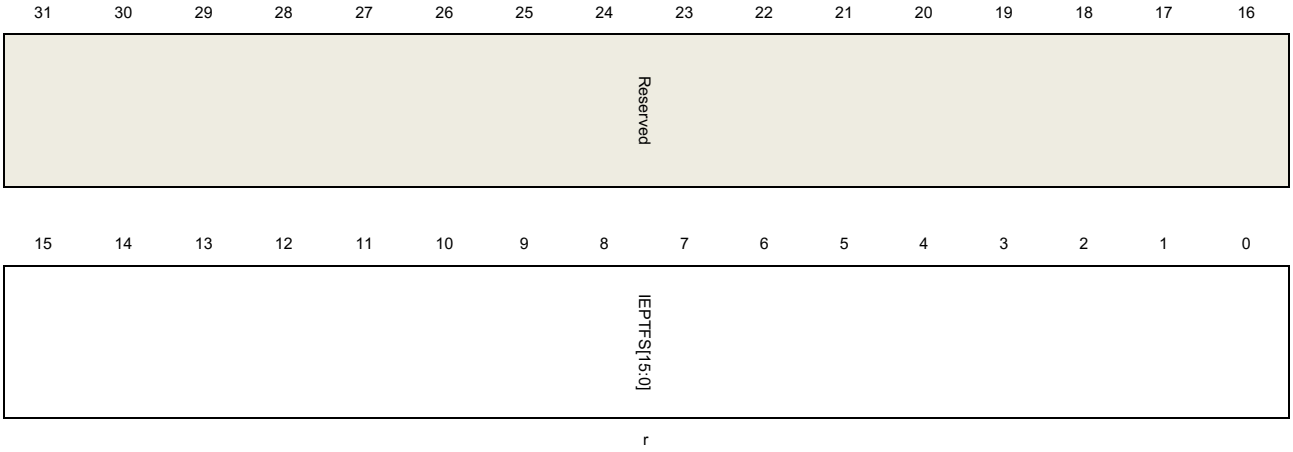
Bits	Fields	Descriptions
31:0	DMAADDR[31:0]	DMA address This field defines the endpoint's DMA address. DMA uses this address to fetch packet data for IN endpoint or write packet data for OUT endpoint.

Device IN endpoint-x transmit FIFO status register (USBHS_DIEPxTFSTAT) (x = 0 .. 5, 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)



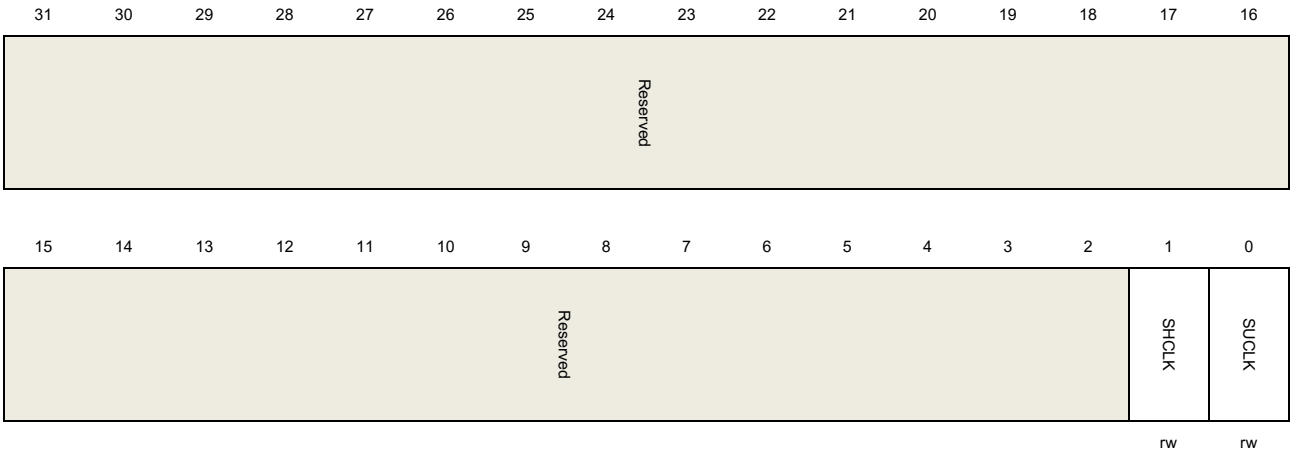
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	IEPTFS[15:0]	IN endpoint's Tx FIFO space remaining IN endpoint's Tx FIFO space remaining in 32-bit word: 0: FIFO is full 1: 1 word available … n: n words available

34.7.4. 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
31:2	Reserved	Must be kept at reset value.
1	SHCLK	Stop HCLK Stop the HCLK to save power. 0: HCLK is not stopped 1:HCLK is stopped
0	SUCLK	Stop the USB clock Stop the USB clock to save power. 0: USB clock is not stopped 1: USB clock is stopped

35. Appendix

35.1. List of abbreviations used in register

Table 35-1. 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)	Software can toggle this bit by writing 1. Writing 0 has no effect.
read-only/set by write 1 (rt_w1)	Software can only read to this bit. Writing 1 trigges the event but has no effect on the bit value.
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.
read/set by read (rs_r)	Software can read this bit, and set this bit by reading. Writing has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing. Writing 0 or 1 has the same effect to this bit.
read-only/write trigger (rt_w)	Software can read this bit. Writing 0 or 1 triggers an event but has no effect on the bit value.

35.2. List of terms

Table 35-2. 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.

Glossary	Descriptions
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.

35.3. Available peripherals

For availability of peripherals and their number across all MCU series types, refer to the corresponding device data datasheet.

36. Revision history

Table 36-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Mar.1, 2024
1.1	1. Updated the description of EBIE and RTIE bits in the USART_CTL3 register. 2. Updated the description of BANK1_Ex in the ***Flash memory architecture*** chapter.	Aug.5, 2024
1.2	1. Update the peripheral interrupt descriptions for IRQ 54 and IRQ 101 in Table 7-2. Interrupt vector table. 2. Remove the description about modifying interrupt vector table offset for booting Bank1 code from bootloader in the ***Boot configuration*** section. 3. Modify the description of the PDR_ON pin in the ***VDD domain*** section. The use of the PDR_ON pin should refer to the recommended circuit diagram in the Datasheet. 4. Modify the bit 15 field name of ***Control register 1 (I2C_CTL1)***. 5. Change DAC0 and DAC1 to DAC0_OUT0 and DAC0_OUT1 in ***Table 14-2. Peripheral requests to DMA0***. 6. Update ***Figure 4-1. Power supply overview***. 7. Update ***Table 18-2. ADC input pins definition***.	Feb.19, 2025
1.3	1. Update the functional description of the SPST bit in the ***Device status register (USBHS_DSTAT)***. 2. Update the description of boot sources in the ***Boot configuration*** section, changing from three boot sources to four boot sources. 3. Update the description of the ES[1:0] bit field in the ***Device status register (USBFS_DSTAT)***. 4. Change the SMBus timeout in section ***24.1.3*** to 25ms. 5. Update the description of the ADDMSK2[2:0] bit field in the ***Slave address register 1 (I2C_SADDR1)*** in section ***24.2.4***. 6. Modify the reset value of the ***Control register 2 (I2C_CTL2)*** to 0x0000FE00. 7. Update ***Figure 5-2. Clock tree***, removing the R coefficient in the PLL and the P coefficient in the PLLI2S. 8. Add a note in the ***I2S clock*** section, requiring the configuration value of the I2S serial clock to be set below 1/6 of the PCLK clock (not including 1/6). 9. Add a note in the ***Error Checking and Correcting (ECC)***. 10. Update the bit fields and descriptions of the ***Input remap register (TIMERx_IRMP) (x=10)***.	Aug.29, 2025

	11. Update the configuration method for Low-driver / Low-power mode in the ***Deep-sleep mode*** section. 12. Add notes regarding CKM in the ***HXTAL clock monitor (CKM)*** section. 13. Add notes regarding CK_OUT in the ***Clock output capability*** section. 14. Add restrictions for using the SMBus Alert function in the ***SMBus alert*** section. 15. Add precautions for operating option bytes in the ***Option bytes modify*** section.	
1.4	1. Update the ***Figure 5-2. Clock tree***. 2. Update the FRI[15:0] bit field description in the ***Host frame interval register (USBHS_HFT)***. 3. Add notes regarding SDADELY=0 in the ***I2C timings configuration*** section. 4. Update the ADDRESS[7:1] bit field description in ***Slave address register 0 (I2C_SADDR0)*** and ***Slave address register 1 (I2C_SADDR1)***. 5. Update the SDADELY[3:0] bit field description in ***Timing register (I2C_TIMING)***. 6. Update the ***Figure 27-1. DCI module block diagram*** and ***Table 27-1. PINs used by DCI***. 7. Add instructions for using SPI CRC. 8. Remove the I023_DRV bit from ***Quad-SPI mode control register (SPI_QCTL) of SPI5***. 9. Update the bit field description of MTF0/1/2 in the ***Transmit status register (CAN_TSTAT)***. 10. Update the description of the valid edge in the ***Bit time*** section and add a note about compliance with the ISO11898 standard. 11. Add the restriction that the ***Timing register (I2C_TIMING)*** cannot be rewritten when I2CEN = 1. 12. Add usage restrictions for the bit fields SADDRESS[9:0], TRDIR, ADD10EN, HEAD10R, and BYTENUM[7:0] in the ***Control register 1 (I2C_CTL1)*** in section ***24.2.4***. 13. Add a note in the ***I2C master mode*** section that BYTENUM[7:0] cannot be configured to zero. 14. Add a note about the RT[23:0] bit field in the ***Receive timeout register (USART_RT)***. 15. Update the description of the division factor in the low-power mode of IrDA in the ***IrDA SIR ENDEC mode*** section.	Jan.20, 2025
1.5	1. Update the calculation formula for the total SDA output delay in the ***I2C timings configuration*** section. 2. Update the description of the CHxCOMFEN bit field in the	Sep.29, 2026

	Channel control register y (TIMERx_CHCTLy). 3. Update the software procedure for using DMA with the ADC in the ***Scan operation mode*** section. 4. Add notes on the OUT transfers operation sequence with DMA disabled in the ***Universal serial bus high-speed interface (USBHS)*** chapter. 5. Update the clearing method of the MSC interrupt status bits, the descriptions of the relevant bit fields in the PTP control register, and the description of the IGBS bit field in the ***Ethernet (ENET)*** chapter. 6. Update the description of the SMBALT bit in the ***Status register (I2C_STAT)***. 7. Update the access attribute of bit 3 in the ***Backup domain control register (RCU_BDCTL)***. 8. Update the ECCEADDR6[31:0] bit field description in the ***FLASH ECC address register (SYSCFG_FLASHECC_ADDR)***. 9. Update the ECCEADDR1[13:0] bit field description of the ***SRAM1 ECC status register (SYSCFG_SRAM1ECC)***.	

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